

LAPORAN AKHIR
SISTEM MANAJEMEN BASIS DATA



Disusun Oleh :

NAMA	: FAIRUZ ABDULLAH
N.R.P	: 220441100070
KELAS	: 4A
DOSEN	: MOHAMMAD SYARIEF, ST., M.Cs
ASISTEN	: NURI HIDAYATULOH



LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

LAPORAN AKHIR PRAKTIKUM SISTEM MANAJEMEN BASIS DATA



Disusun Oleh :

NAMA	: FAIRUZ ABDULLAH
N.R.P	: 220441100070
KELAS	: 4A
DOSEN	: MOHAMMAD SYARIEF, ST., M.Cs
ASISTEN	: NURI HIDAYATULOH

Disetujui : 14 JUNI 2024
Asisten

NURI HIDAYATULOH
210441100100



LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

KATA PENGANTAR

Dengan memanjatkan puji syukur kepada Allah SWT, yang telah memberikan rahmat serta hidayah-Nya, sehingga laporan akhir praktikum Sistem Manajemen Basis Data (SMBD) ini dapat diselesaikan dengan baik dan tepat waktu. Laporan ini merupakan hasil dari kerja keras dan dedikasi selama mengikuti praktikum pada semester satu, yang diharapkan dapat memberikan manfaat bagi pembaca dalam memahami konsep-konsep dasar dari SMBD.

Penyusun mengucapkan terima kasih yang sebesar-besarnya kepada dosen SMBD, Bapak MOHAMMAD SYARIEF, ST., M.Cs, yang telah memberikan bimbingan dan ilmu dari awal hingga akhir perkuliahan. Materi yang beliau sampaikan sangat membantu dalam penyusunan laporan ini, serta memperdalam pemahaman saya tentang SMBD. Ucapan terima kasih juga penyusun sampaikan kepada rekan-rekan asisten praktikum yang selalu siap membantu dan memberikan arahan selama pelaksanaan praktikum. Bantuan dan dukungan mereka sangat berharga dalam menyelesaikan tugas-tugas praktikum serta memahami alur kerja dalam pengelolaan basis data.

Tidak lupa, penyusun juga mengucapkan terima kasih kepada teman-teman seperjuangan yang telah memberikan dukungan dan bantuan selama proses penyusunan laporan ini. Kerjasama dan diskusi yang kami lakukan bersama sangat membantu dalam memperkaya wawasan dan memperbaiki kesalahan yang ada. Penyusun menyadari bahwa laporan ini masih jauh dari sempurna dan mungkin masih terdapat kekurangan serta kesalahan. Oleh karena itu, penyusun mohon maaf jika terdapat kekurangan dalam laporan ini dan sangat mengharapkan kritik serta saran yang membangun untuk perbaikan di masa yang akan datang.

Demikian yang dapat penyusun sampaikan terkait laporan ini. Semoga laporan ini dapat memberikan manfaat dan pengetahuan tambahan bagi para pembaca. Terima kasih.

Bangkalan, 19 JUNI 2024

Penyusun

FAIRUZ ABDULLAH

NRP. 220441100070



LEMBAR ASISTENSI
PRAKTIKUM SISTEM MANAJEMEN BASIS DATA
FAKULTAS TEKNIK PRODI SISTEM INFORMASI
UNIVERSITAS TRUNOJOYO MADURA



Nama : FAIRUZ ABDULLAH

NIM : 220441100070

Kelas : SISTEM MANAJEMEN BASIS DATA 4A



No	Materi	Tanggal Praktikum	Tanggal Asistensi	Catatan Asisten	TTD
1	Data Definition Language (DDL) dan DML	22-03-2024	27-03-2024		
2	VIEW	29-03-2024	30-04-2024		
3	Stored Procedure	26-04-2024	02-05-2024		
4	SQL	17-05-2024	16-05-2024		
5	Type data	17-05-2024	16-05-2024		
6	stored procedure : Branching/looping	03-05-2024	16-05-2024		
7	Trigger	24-05-2024	10-06-2024		

Bangkalan,

2024

Asisten Praktikum

NURI HIDAYATULLOH

NIM. 210441100100

LAPORAN RESMI
MODUL I
DATA DEFINITION LANGUAGE (DDL) DAN
DATA MANIPULATION LANGUAGE (DML)
SISTEM MANAJEMEN BASIS DATA



NAMA	: FAIRUZ ABDULLAH
N.R.P	: 220441100070
DOSEN	: MOHAMMAD SYARIEF, ST., M.Cs
ASISTEN	: NURI HIDAYATULOH
TGL PRAKTIKUM	: 22 MARET 2024

Disetujui : 27 Maret 2024
Asisten

NURI HIDAYATULOH
210441100100



LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam era digital yang semakin berkembang pesat seperti saat ini, pengelolaan data telah menjadi aspek krusial dalam berbagai bidang, termasuk bisnis, industri, pendidikan, dan pemerintahan. Data yang tersimpan dan dikelola dengan baik memegang peranan penting dalam pengambilan keputusan yang tepat dan berkelanjutan. Oleh karena itu, diperlukan sistem yang dapat mengatur dan mengelola data secara efisien, dan di sinilah Sistem Manajemen Basis Data (DBMS) berperan. DBMS memungkinkan organisasi untuk menyimpan, mengelola, dan mengakses data dengan cara yang terstruktur dan terorganisir, serta menyediakan alat-alat yang diperlukan untuk memanipulasi data sesuai kebutuhan.

Kehadiran DBMS tidak hanya mempermudah akses dan pengelolaan data, tetapi juga memberikan fleksibilitas dalam mengatur struktur data sesuai dengan kebutuhan spesifik dari suatu aplikasi atau sistem. Dua komponen utama dalam DBMS yang memainkan peran penting dalam manajemen data adalah Data Definition Language (DDL) dan Data Manipulation Language (DML). DDL digunakan untuk mendefinisikan struktur database, termasuk pembuatan tabel, definisi kolom, dan pengaturan kunci primer dan kunci asing, sementara DML digunakan untuk memanipulasi data yang tersimpan di dalam database, seperti mengambil, menambahkan, mengubah, atau menghapus data. Oleh karena itu, pemahaman yang baik tentang DDL dan DML sangat penting bagi siapa pun yang terlibat dalam pengelolaan dan penggunaan basis data.

1.2 Tujuan

- Mampu memahami perintah-perintah untuk menjelaskan objek dari database dan mendefinisikan atribut-atribut database, table dan batasan-batasan terhadap suatu atribut serta hubungan antar table
- Mampu memahami dan mengisi table dalam database dan memanipulasi data

BAB II

DASAR TEORI

2.1 Data Definition Language

DDL SQL memungkinkan dilakukannya spesifikasi tidak hanya pada himpunan relasi tetapi juga informasi untuk setiap relasi, yang meliputi :

- Skema setiap relasi
- Domain nilai setiap atribut relasi
- Konstrain integritas
- Himpunan indeks untuk setiap relasi
- Sekuriti dan otorisasi setiap relasi
- Struktur penyimpanan secara fisik untuk setiap relasi dalam disk.

DDL (Data Definition Language) merupakan sekumpulan set perintah yang bertujuan untuk mendefinisikan atribut – atribut database, tabel, atribut kolom (field), maupun batasan – batasan terhadap suatu atribut dan relasi/hubungan antar tabel. Yang termasuk dalam kelompok perintah DDL adalah : CREATE, ALTER, dan DROP.

CREATE merupakan perintah DDL yang digunakan untuk membuat database maupun tabel. Nama database maupun tabel tidak boleh mengandung spasi (space). Nama database tidak boleh sama antar database.

ALTER merupakan perintah DDL yang digunakan untuk mengubah nama/struktur tabel. **DROP** merupakan perintah DDL yang digunakan untuk menghapus database ataupun tabel.

2.1.1 Database

Kalo variabel adalah tempat yang kita gunakan untuk menyimpan berbagai macam tipe data, berarti tipe data adalah jenis data yang disimpan di dalam variabel, contohnya kaya gambar di bawah ini:

Database dapat dibayangkan sebagai sebuah lemari arsip, dimana arsip yang disimpan didalam lemari tersebut akan disusun dan dikelompokkan berdasarkan urutan tertentu seperti urutan abjad atau urutan kronologis. Hal ini

dilakukan sebagai upaya untuk mempermudah pencarian arsip, karena arsip yang tersusun dengan rapi maka proses pencarian arsip dapat lebih cepat.

Syntax untuk membuat database sebagai berikut :

```
CREATE DATABASE namadatabase;
```

Sedangkan syntax tambahan untuk menampilkan daftar nama database yang terdapat dalam database server pada MySQL menggunakan perintah :

```
SHOW DATABASES;
```

Sebelum kita membuat suatu tabel yang digunakan untuk menyimpan data, terlebih dahulu harus memilih/mengaktifkan salah satu database sebagai database aktif yang akan digunakan untuk menyimpan beberapa tabel yang akan kita buat.

```
USE namadatabase;
```

Untuk memilih/mengaktifkan salah satu database menggunakan syntax :

Contohnya kita akan mengaktifkan database db_universitas : `USE db_universitas;` Selain perintah - perintah di atas, terdapat perintah yang berfungsi untuk menghapus database maupun tabel. Perintah tersebut adalah sebagai berikut :

```
DROP DATABASE namadatabase;
```

2.1.2 Tabel

Dalam membuat sebuah tabel, nama tabel tidak boleh mengandung spasi (space). Ketika membuat tabel ada beberapa yang harus dideklarasikan dalam pembuatannya, yaitu antara lain meliputi :

✓ Nama tabel,

✓

□

Adapun syntax yang digunakan untuk membuat tabel secara umum adalah sebagai berikut:

```
CREATE TABLE namatabel (Field1, TypeData1, Field2,  
                          TypeData2);
```

Contoh berikut syntax untuk membuat Tabel mahasiswa:

```
CREATE TABLE mahasiswa (nim CHAR (20), nama_mhs CHAR (50),  
login CHAR(20), pass CHAR(20), umur INT, ipk real, PRIMARY  
KEY(nim));
```

- Menampilkan Tabel

Untuk menampilkan daftar nama tabel yang terdapat dalam database yang sedang aktif/digunakan menggunakan perintah :

```
SHOW TABLES;
```

- Menampilkan deskripsi atribut table

Untuk menampilkan deskripsi atribut – atribut yang terdapat pada suatu tabel dengan menggunakan perintah:

```
DESC namaTabel;
```

- Menghapus table

Untuk menghapus Tabel perintahnya sama dengan untuk menghapus database yaitu dengan menggunakan perintah DROP. Syntax yang digunakan adalah:

Tabel yang akan dihapus harus sesuai dengan nama tabel. Misal kita akan menghapus tabel mahasiswa, maka syntax nya adalah: DROP TABLE mahasiswa;

- Mendefinisikan Null/Not Null

Null ataupun Not Null merupakan pernyataan yang digunakan untuk membuat kolom yang kita buat boleh kosong (Null) atau tidak boleh kosong (Not Null). Ketika pada kolom tabel tidak diset, maka secara default akan bernilai Null (boleh kosong).

Untuk mendefinisikannya maka perintah yang digunakan adalah:

```
CREATE TABLE mahasiswa (nim CHAR (20) NOT NULL, nama_mhs  
CHAR (50) NOT  
NULL,login CHAR(20) NOT NULL, pass CHAR(20) NOT NULL, umur  
INT, ipk real, PRIMARY KEY(nim));
```

- Mendefinisikan Nilai Default

Nilai default merupakan nilai yang diberikan secara otomatis oleh sistem untuk suatu kolom ketika terjadi penambahan baris baru, sementara nilai pada kolom tersebut tidak diisi oleh pengguna.Contohnya adalah :

```
CREATE TABLE mahasiswa (nim CHAR (20), nama_mhs CHAR (50),  
login CHAR(20), pass CHAR(20), umur INT DEFAULT 0, ipk real,  
PRIMARY KEY(nim));
```

- Mendefinisikan PRIMARY KEY pada Tabel

Suatu keharusan dalam suatu tabel adalah harus memiliki satu kolom yang dijadikan sebagai perwakilan dari tabel tersebut. Pembuatan perwakilan tabel ini berfungsi untuk melakukan hubungan/relasional dengan tabel lain. Bentuk perwakilan ini dalam database disebut sebagai PRIMARY KEY yang aturan pembuatannya adalah sebagai berikut:

- Satu tabel hanya diperbolehkan memiliki satu kolom kunci.
- Nama kolom kunci tidak digunakan pada kolom lain dalam satu tabel
- Nama kolom kunci tidak boleh sama dengan kolom kunci yang ada pada tabel

Terdapat tiga cara untuk mendefinisikan primary key. Berikut ini syntax yang digunakan:

```
CREATE TABLE mahasiswa (nim CHAR (20), nama_mhs CHAR  
(50), login CHAR(20), pass CHAR(20), umur INT, ipk real, PRIMARY  
KEY(nim));
```

Atau

```
CREATE TABLE mahasiswa (nim CHAR (20) NOT NULL PRIMARY  
KEY, nama_mhs CHAR (50), login CHAR(20), pass CHAR(20), umur INT,  
ipk real);
```

Atau

```
ALTER TABLE mahasiswa ADD CONSTRAINT namaconstraint PRIMARY  
KEY(namakolom);
```

- Menghapus PRIMARY KEY pada Tabel

Cara 1: Jika primary key dibuat menggunakan alter table:

```
ALTER TABLE namatabel DROP CONSTRAINT namaconstraint;
```

Cara 2: jika primary key dibuat melalui create table:

```
ALTER TABLE namatabel DROP PRIMARY KEY;
```

- Menambah kolom baru pada tabel

Pada saat kita membuat tabel terkadang kita ingin menambahkan kolom lagi pada tabel yang sudah kita buat. Dalam database, hal ini dapat dilakukan dengan menggunakan perintah sebagai berikut:

```
ALTER TABLE namatabel ADD fieldbaru typedata(lebar);
```

Namatabel merupakan nama tabel yang akan ditambahkan kolomnya. Field baru merupakan nama kolom yang akan ditambahkan, typedata(lebar) merupakan type data dan lebar data yang akan ditambahkan.

Misal kita akan menambahkan kolom telepon pada tabel mahasiswa setelah

```
ALTER TABLE mahasiswa ADD COLUMN telepon CHAR(15)
AFTERumur;
```

- Mengubah Tipe Data atau Lebar Kolom pada TabelPerintah yang digunakan adalah:

```
ALTER TABLE namatabel MODIFY COLUMN field
```

Contoh :

```
ALTER TABLE mahasiswa MODIFY COLUMN CHANGE
COLUMN
telepon(12);
```

- ☐ Mengubah Nama Kolom (Field) Perintah

```
ALTER TABLE namatabel namakolomlama namakolombaru
typedatabaru(lebarbaru);
```

yang digunakan adalah:

Contoh :

```
ALTER TABLE mahasiswa CHANGE COLUMN telepon phone
CHAR(25);
```

- ☐ Menghapus Kolom pada Tabel Perintah yang digunakan:

```
ALTER TABLE namatabel DROP COLUMN namakolom;
```

Contoh :

```
ALTER TABLE mahasiswa DROP COLUMN phone;
```

☐ Membuat dan Menghapus Index

Index berfungsi untuk mempercepat proses pencarian data dalam suatu tabel.

tabel. Terdapat perintah untuk membuat dan menghapus index, tapi tidak ada perintah untuk merubah index. Perhatikan contoh berikut:

```
CREATE INDEX IDXNIM ON mahasiswa(nim);
```

Atau

```
ALTER TABLE mahasiswa ADD INDEX IDXNIM(nim);
```

Sedangkan untuk menghapus :

```
DROP INDEX IDXNIM ON mahasiswa(nim);
```

Atau

```
ALTER TABLE mahasiswa DROP INDEX IDXNIM(nim);
```

2.2 Data Manipulation Language

Merupakan bentuk bahasa basis data yang berguna untuk melakukan manipulasi data dan pengambilan data pada suatu basis data. Manipulasi data dapat berupa :

- a. penyisipan/penambahan data baru ke suatu basis data
- b. penghapusan data dari suatu basis data
- c. pengubahan data di suatu basis data

Pada level fisik, kita harus mendefinisikan algoritma yang memungkinkan pengaksesan yang efisien terhadap data. Pada level yang lebih tinggi, yang dipentingkan bukan hanya efisiensi akses, tetapi juga efisiensi interaksi

memudahkan pemakai untuk mengakses data sebagaimana direpresentasikan oleh model data. Ada 2 jenis DML, yaitu :

- Prosedural, yang mensyaratkan agar pemakai menentukan, data apa yang diinginkan serta bagaimana cara mendapatkannya.
- Nonprosedural, yang membuat pemakai dapat menentukan data apa yang diinginkan tanpa menyebutkan bagaimana cara mendapatkannya.

Perintah yang termasuk dalam DML adalah: INSERT, DELETE, UPDATE, dan SELECT

2.2.1 Insert

Perintah INSERT bertujuan untuk menambahkan record data pada suatu tabel.

Terdapat beberapa cara untuk menambahkan record, yaitu:

Cara 1: Menambahkan record dengan mengisi data pada setiap kolom:

INSERT INTO namatabel VALUES (nilai1, nilai2, nilai-

Cara 2: menambahkan baris dengan hanya mengisi pada kolom tertentu:

```
INSERT INTO namatabel (field1, field2, field-n) VALUES
```

```
(nilai1, nilai2, nilai-n);
```

Ket : Jika data bertipe string, date, atau time (contoh : didi, basis data, 1984-03-18) maka pemberian nilainya diapit menggunakan tanda petik tunggal ('Didi') atau petik ganda ("Basis Data"). Jika data bertipe numerik (29, 4)

maka pemberian nilainya tidak diapit tanda petik tunggal maupun ganda.

2.2.2 Delete

Perintah DELETE digunakan untuk menghapus satu baris, baris dengan kondisi tertentu maupun seluruh baris. Syntax yang digunakan:

DELETE FROM namatabel WHERE [kondisi];
--

Perintah dalam tanda [] bersifat pilihan/opsional untuk menghapus suatu baris dengan kondisi tertentu yang dipersyaratkan. Contoh perintah untuk menghapus suatu baris dalam tabel dengan kondisi persyaratan tertentu :

DELETE FROM mahasiswa WHERE nim 13120070;

2.2.3 Update

Perintah UPDATE digunakan untuk mengubah isi data pada satu atau beberapa kolom pada suatu tabel. Syntax yang digunakan secara umum adalah sebagai berikut:

UPDATE namatabel SET field1=nilai1, field2=nilai2

Perintah dalam tanda [] bersifat pilihan/opsional untuk mengubah suatu baris dengan kondisi tertentu yang dipersyaratkan.

2.2.4 SELECT

Perintah SELECT digunakan untuk menampilkan isi dari suatu tabel yang dapat dihubungkan dengan beberapa tabel lainnya.

Menampilkan data semua kolom dengan menggunakan asterisk (*):

```
SELECT * FROM namatabel;
```

Menampilkan data untuk field/kolom tertentu:

```
SELECT field1, field2, field-n FROM namatabel;
```

Menampilkan data dengan kondisi tertentu menggunakan klausa WHERE:

```
SELECT * FROM namatabel WHERE kondisi;
```

BAB III

TUGAS PENDAHULUAN

3.1 Soal

1. Apa yang anda ketahui tentang DDL?
2. Apa saja yang termasuk Perintah DDL? berikan contoh programnya!
3. Apa yang anda ketahui tentang DML?
4. Apa saja yang termasuk perintah DML? berikan contoh programnya!
5. Tuliskan Query untuk membuat tabel dengan nama DATA_PENDUDUK dengan kolom (NIK, NAMA LENGKAP, ALAMAT, TANGGAL LAHIR), untuk tipe data dan panjang data tersebut, kemudian tambahkan 3 baris data pada tabel PENDUDUK tadi (input data tersebut)

3.2 Jawaban

1. DDL (Data Definition Language) adalah bagian dari bahasa SQL yang digunakan untuk mendefinisikan struktur dan karakteristik dari objek database
2. • CREATE : Digunakan untuk membuat objek baru
• ALTER : Digunakan untuk mengubah struktur objek yang sudah ada pada database
• DROP : Digunakan untuk menghapus objek

Contoh Program menggunakan DDL:

```
CREATE TABLE Mahasiswa (  
    NIM INT PRIMARY KEY,  
    Nama VARCHAR(100),  
    Jurusan VARCHAR(50)  
);
```

```
DROP TABLE Mahasiswa;
```

3. DML (Data Manipulation Language) adalah bagian dari bahasa SQL yang digunakan untuk memanipulasi data dalam database

4. INSERT: menambah data baru ke dalam tabel

UPDATE: mengubah data yang sudah ada dalam tabel

DELETE: menghapus data dari tabel

Contoh:

```
INSERT INTO Mahasiswa (NIM, NAMA, Prodi)
```

```
VALUES (123456, 'Joko Doe', 'Informatika');
```

```
UPDATE Mahasiswa SET Program Studi = 'Sistem Informasi' where nim = 123456;
```

5. CREATE table Data_Penduduk (

NIK Varchar (20) Primary Key,

NAMA Varchar (100),

ALAMAT Varchar (100),

TANGGAL_LAHIR Date

);

```
INSERT into Data_Penduduk (NIK, Nama, Alamat,
```

```
tanggal_lahir)
```

```
VALUES ('1234567890', 'Joko Doe', 'Jl Raya', '1990-05-15'
```

```
('987654321', 'Jane Smith', 'Jl Mawar', '1985-10-20'
```

```
('2345678901', 'Michael Sohn', 'Jl Anggrek', '1989-02-28'
```

```
);
```

BAB IV

IMPLEMENTASI

4.1 Soal

1. Buatlah database tentang sebuah toko swalayan ingin mengembangkan database untuk mencatat transaksi yang terjadi.

☐ Transaksi berupa :

- POS / Penjualan : ke pelanggan
- Pengembalian barang / retur
- Stok
- Pembelian barang dari supplier
- Koreksi stok

☐ Keuangan berupa :

- Penjualan, masuk ke akun penjualan
- Pembelian dari supplier, masuk ke akun : hutang (pembayaran ke supplier, mengurangi akun hutang)

☐ Laporan berupa :

- Penjualan
- Stok

2. Isi data pada tabel masing – masing 10 data

3. Tampilkan keseluruhan data pada setiap tabel.

4. Lakukan perubahan pada salah satu nama tabel

5. Hapus database tersebut.

4.2 Source Code

```
CREATE DATABASE toko_swalayan;USE toko_swalayan;

CREATE TABLE Barang ( ID_Barang INT PRIMARY KEY
AUTO_INCREMENT,
    Nama_Barang VARCHAR(100),Harga DECIMAL(10, 2),
    Jumlah_Stok INT
);

CREATE TABLE Transaksi ( ID_Transaksi INT PRIMARY KEY
AUTO_INCREMENT,
    Jenis_Transaksi VARCHAR(50),Waktu_Transaksi DATE,
    Total_Harga DECIMAL(10, 2), Keterangan VARCHAR(100)
);

CREATE TABLE Supplier (
    ID_Supplier INT PRIMARY KEY AUTO_INCREMENT,
    Nama_Supplier VARCHAR(100),Alamat VARCHAR(255),
    No_Telepon VARCHAR(20)
);

CREATE TABLE Laporan (
    ID_Laporan INT PRIMARY KEY AUTO_INCREMENT,
    Tanggal_Laporan DATE, Jenis_Laporan VARCHAR(100),
    Keterangan VARCHAR(100)
```

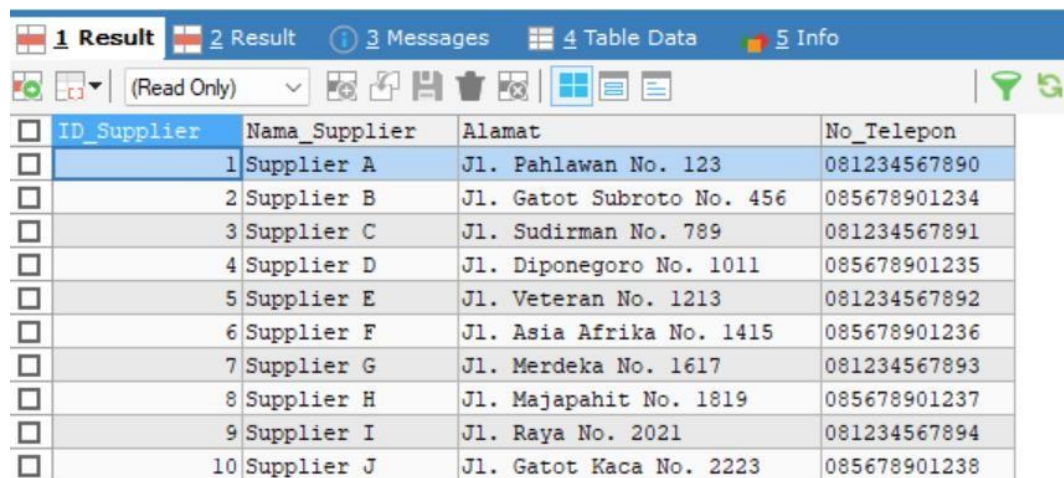
```
INSERT INTO Barang (Nama_Barang, Harga, Jumlah_Stok)
VALUES
```

```
('Sabun Mandi', 5000.00, 100),
('Shampoo', 10000.00, 50),
('Pasta Gigi', 8000.00, 80),
('Tisu Basah', 3000.00, 120),
('Minyak Goreng', 15000.00, 200),
('Gula Pasir', 12000.00, 150),
('Kopi Bubuk', 25000.00, 70),
('Teh Celup', 18000.00, 90),
('Beras', 30000.00, 100),
('Mie Instan', 5000.00, 120);
```

```
INSERT INTO Transaksi (Jenis_Transaksi, Waktu_Transaksi,
Total_Harga, Keterangan) VALUES
```

```
('POS / Penjualan : ke pelanggan', '2024-03-25 ', 25000.00,
'Transaksi penjualan barang A'),
('POS / Penjualan : ke pelanggan', '2024-03-25 ', 18000.00,
'Transaksi penjualan barang B'),
('Pengembalian barang / retur', '2024-03-25 ', 15000.00,
'Pengembalian barang C karena cacat'),
('Pengembalian barang / retur', '2024-03-26 ', 12000.00,
'Pengembalian barang D karena tidak sesuai pesanan'),
('Stok', '2024-03-26 ', NULL, 'Penyesuaian stok barang E
setelah inventarisasi'),
('Stok', '2024-03-27 ', NULL, 'Penyesuaian stok barang F
setelah pengecekan kualitas'),
('Pembelian barang dari supplier', '2024-03-27 ', 35000.00,
'Pembelian barang G dari Supplier J'),
('Pembelian barang dari supplier', '2024-03-28 ', 28000.00,
'Pembelian barang H dari Supplier I'),
```

4.3 Hasil



ID_Supplier	Nama_Supplier	Alamat	No_Telepon
1	Supplier A	Jl. Pahlawan No. 123	081234567890
2	Supplier B	Jl. Gatot Subroto No. 456	085678901234
3	Supplier C	Jl. Sudirman No. 789	081234567891
4	Supplier D	Jl. Diponegoro No. 1011	085678901235
5	Supplier E	Jl. Veteran No. 1213	081234567892
6	Supplier F	Jl. Asia Afrika No. 1415	085678901236
7	Supplier G	Jl. Merdeka No. 1617	081234567893
8	Supplier H	Jl. Majapahit No. 1819	085678901237
9	Supplier I	Jl. Raya No. 2021	081234567894
10	Supplier J	Jl. Gatot Kaca No. 2223	085678901238

4.4 Penjelasan

- Membuat Database: Baris pertama (CREATE DATABASE toko_swalayan;) membuat database baru dengan nama toko_swalayan yang akan digunakan untuk menyimpan semua tabel.
- Menggunakan Database: Baris kedua (USE toko_swalayan;) menginstruksikan server database untuk menggunakan database toko_swalayan yang telah dibuat.
- Membuat Tabel Barang: Berikutnya, dibuat tabel Barang dengan kolom ID_Barang, Nama_Barang, Harga, dan Jumlah_Stok. Ini akan digunakan untuk menyimpan informasi tentang barang-barang yang dijual di toko swalayan.
- Membuat Tabel Transaksi: Tabel Transaksi memiliki kolom ID_Transaksi, Jenis_Transaksi, Waktu_Transaksi, Total_Harga, dan Keterangan. Ini akan digunakan untuk merekam semua transaksi yang terjadi di toko swalayan.
- Membuat Tabel Supplier: Tabel Supplier memiliki kolom ID_Supplier, Nama_Supplier, Alamat, dan No_Telepon. Ini akan digunakan untuk menyimpan informasi tentang pemasok barang kepada toko swalayan.
- Membuat Tabel Laporan: Tabel Laporan memiliki kolom ID_Laporan, Tanggal_Laporan, Jenis_Laporan, dan Keterangan. Ini akan digunakan untuk merekam semua laporan yang berkaitan dengan operasi toko swalayan.

BAB V

PENUTUP

5.1 Analisa

Praktikan menganalisis bahwa pemahaman yang mendalam tentang Data Definition Language (DDL) dan Data Manipulation Language (DML) menjadi landasan yang krusial dalam mengelola basis data secara efektif. DDL memungkinkan praktikan untuk merancang struktur basis data dengan tepat, termasuk pembuatan tabel, kolom, kunci, dan konstrain yang diperlukan. Kemampuan untuk menggunakan DDL dengan baik memastikan bahwa basis data dapat dibentuk sesuai dengan kebutuhan aplikasi atau sistem yang diimplementasikan.

Di sisi lain, DML memberikan akses kepada praktikan untuk memanipulasi data yang ada dalam basis data. Dengan perintah seperti SELECT, INSERT, UPDATE, dan DELETE, mereka dapat mengambil, menambahkan, mengubah, dan menghapus data sesuai dengan kebutuhan. Pemahaman yang baik tentang DML memungkinkan praktikan untuk melakukan operasi-operasi ini dengan akurat dan efisien, memastikan integritas dan konsistensi data dalam basis data. Dengan demikian, pemahaman tentang DDL dan DML menjadi esensial bagi siapa pun yang terlibat dalam pengelolaan dan penggunaan basis data.

5.2 Kesimpulan

1. Pentingnya Pemahaman DDL dan DML, pemahaman yang baik tentang Data Definition Language (DDL) dan Data Manipulation Language (DML) adalah fondasi yang krusial dalam mengelola basis data dengan efektif.
 2. Peran Utama dalam Pengelolaan Basis Data, pengetahuan tentang DDL dan DML tidak hanya penting bagi administrator basis data, tetapi juga bagi pengembang perangkat lunak, analis data, dan pemangku kepentingan lain yang terlibat dalam penggunaan basis data.
-

LAPORAN RESMI
MODUL II
VIEW
SISTEM MANAJEMEN BASIS DATA



NAMA	: FAIRUZ ABDULLAH
N.R.P	: 220441100070
DOSEN	: MOHAMMAD SYARIEF, ST., MCs
ASISTEN	: NURI HIDAYATULOH
TGL PRAKTIKUM	: 29 MARET 2024

Disetujui : 30 APRIL 2024
Asisten

NURI HIDAYATULOH
210441100100



LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam praktikum sistem manajemen basis data (SMBD), penggunaan View menjadi salah satu konsep penting yang dipelajari. View merupakan objek virtual yang memungkinkan pengguna untuk melihat dan mengakses data dari satu atau lebih tabel dalam basis data tanpa perlu menyimpan data secara fisik. Penggunaan View dalam praktikum SMBD sering kali difokuskan pada beberapa tujuan utama.

View digunakan untuk menyederhanakan analisis data dengan menciptakan tampilan yang terkait dari beberapa tabel, memudahkan proses analisis data yang kompleks. Selain itu, penggunaan View juga membantu dalam mengabstraksi data dengan menyediakan tampilan yang terstruktur dan terfokus, sehingga memudahkan pengguna dalam memahami dan mengelola data.

View juga dapat digunakan untuk tujuan keamanan data, di mana penggunaan View dapat membantu mengontrol akses pengguna terhadap data dengan membatasi tampilan data yang dapat dilihat oleh pengguna tertentu, sesuai dengan aturan dan kebijakan keamanan yang ditetapkan. Dengan demikian, penggunaan View dalam praktikum SMBD memberikan manfaat besar dalam mempermudah analisis data, mengabstraksi kompleksitas data, dan menjaga keamanan data sesuai dengan kebutuhan pengguna.

1.2 Tujuan

- Mampu Memahami apa itu view pada penggunaan Sistem Manajemen Basis Data
 - Mampu memahami perintah-perintah untuk menjelaskan view dari database dan mendefinisikan atribut-atribut view , table dan batasan-batasan terhadap suatu atribut serta hubungan antar table dalam view.
 - Mampu mempraktikkan query view dan membuat View dengan query
 - Memahami pembelajaran tentang Sistem Manajemen Basis Data Secara Utuh dan Menyeluruh
 - Mampu Mempraktikkan Query Sederhana pada Pemabahasan Awal View
-

BAB II

DASAR TEORI

2.1 View

View adalah tabel virtual yang terbuat dari suatu query terhadap satu tabel atau beberapa tabel. View tidak ada secara nyata dalam database. View hanya digunakan untuk menyederhanakan dan mempermudah persepsi pengguna dalam database. Tidak seperti pada umumnya tabel di dalam basis data relasional, view bukanlah bagian dari skema fisik. View bersifat dinamis, ia mengandung data dari tabel yang direpresentasikannya. Dengan demikian, ketika tabel yang menjadi sumber datanya berubah, data di view juga akan berubah. Kegunaan view antara lain:

1. Memfokuskan pada data tertentu
2. Penyederhanaan manipulasi data
3. Menyesuaikan data dengan kebutuhan user
4. Export dan impor data
5. Mengkombinasikan data terpartisiataupun tabel.

Syntax

```
CREATE VIEW view_name [(column[,...n])] [with encryption]
AS select_statement [with check option]
```

Contoh : Buatlah view untuk membuat daftar seluruh nama kota yang ada dalam tabel PLAYERS!

```
CREATE VIEW TOWNS (TOWN) AS
SELECT DISTINCT TOWN
FROM PLAYERS
```

1. Updatetable View

View dapat berisi read-only atau updatable. Kondisi ini sangat dipengaruhi oleh adanya pendefinisian view itu sendiri. Bagaimanapun, untuk menciptakan updatable view, pernyataan SELECT yang didefinisikan di view harus mengikuti aturan-aturan sebagai berikut:

1. Pernyataan SELECT tidak boleh merujuk ke lebih dari satu tabel.SELECT
2. Pernyataan SELECT tidak boleh menggunakan klausa GROUP BY atau HAVING.
3. Pernyataan SELECT harus tidak menggunakan DISTINCT.
4. Pernyataan SELECT harus tidak merujuk ke view lain yang tidak updatable.
5. Pernyataan SELECT tidak boleh mengandung ekspresi apa pun, misalnya fungsi agregat.

Pada hakikatnya, jika sistem database mampu menentukan pemetaan balik dari skema view ke skema tabel dasar, maka view memungkinkan untuk di update. Dalam kondisi ini, operasi-operasi INSERT, UPDATE dan DELETE dapat diterapkan pada view.

2.3 Hapus View

Suatu view dari query selalu menampilkan data yang terbaru, sebagai contoh bila kita memodifikasi sebagian tupel dalam tabel dasarnya dimana view tersebut didefinisikan maka secara otomatis akan berpengaruh pada view di query. Jika tidak membutuhkan view lagi, kita bisa menggunakan perintah

Syntax

```
DROP VIEW view_name
```

BAB III

TUGAS PENDAHULUAN

3.1 Soal

1. Jelaskan apa pengertian dari View!
2. Apa yang terjadi ketika tabel yang asli dihapus?
3. Jelaskan query dibawah ini!

```
Create View TODOList AS  
Select Tasks TaskName, Tasks Description  
From status Inner Join  
    Task on status, statusID = Tasks, statusID  
Where (status.statusID = 1)
```

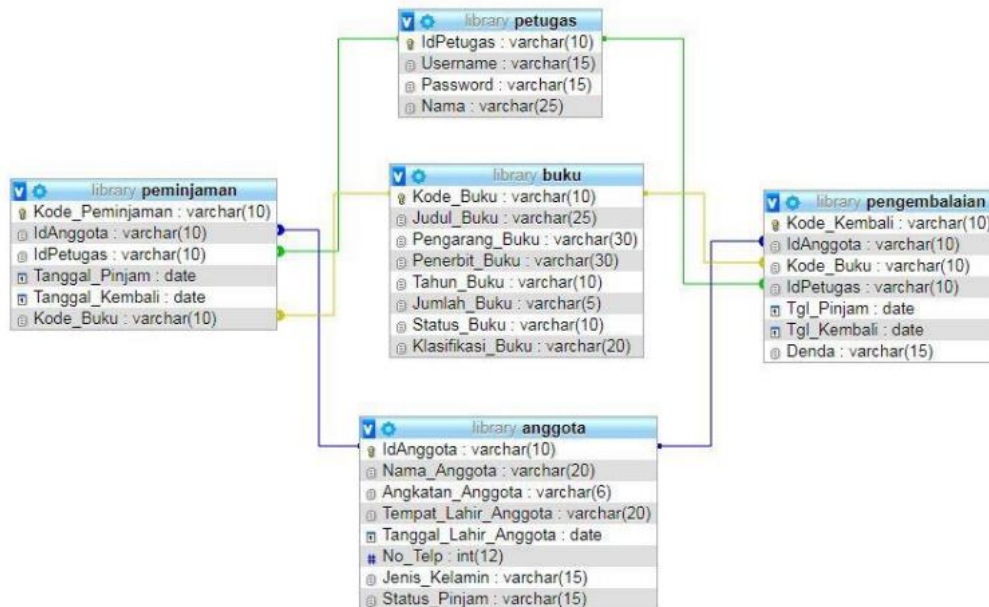
3.2 Jawaban

1. View adalah sebuah objek dalam database yang mempresentasikan kumpulan data yang tersimpan dalam database dengan cara yang ditentukan
2. Ketika tabel asli dihapus, view yang mengacu pada tabel tersebut akan menjadi tidak valid atau broken
3. Perintah untuk membuat sebuah view bernama TODOList, view ini akan menampilkan tugas dan deskripsi tugas dari tabel Tasks yang memiliki Id = 1

BAB IV

IMPLEMENTASI

4.1 Soal



1. Definisikan view untuk mendapatkan data anggota yang hanya meminjam buku lebih dari 5 buku!
2. Definisikan view petugas yang melakukan transaksi (peminjaman/pengembalian) beserta jumlah buku yang terpinjam Ketika petugas tersebut bertugas!
3. Definisikan view dari tugas praktikum nomor 2 hanya petugas yang melayani transaksi terbanyak!
4. Definiskan view buku yang terpinjam paling banyak!

4.2 Source Code

```
CREATE TABLE IF NOT EXISTS anggota (  
  id_anggota VARCHAR(10) NOT NULL PRIMARY KEY,  
  nama_anggota VARCHAR(20) NOT NULL,  
  tempat_lahir_anggota VARCHAR(20) NOT NULL,  
  tanggal_lahir_anggota DATE,  
  no_telp INT(12) NOT NULL,  
  jenis_kelamin VARCHAR(15) NOT NULL  
);  
CREATE TABLE IF NOT EXISTS petugas (  
  id_petugas VARCHAR(10) NOT NULL PRIMARY KEY,  
  username VARCHAR(15) NOT NULL,  
  PASSWORD VARCHAR(15) NOT NULL,  
  nama VARCHAR(25) NOT NULL  
);  
CREATE TABLE IF NOT EXISTS buku (  
  kode_buku VARCHAR(10) NOT NULL PRIMARY KEY ,  
  judul_buku VARCHAR(25) NOT NULL,  
  pengarang_buku VARCHAR(30) NOT NULL,  
  tahun_buku VARCHAR(10) NOT NULL,  
  jumlah_buku VARCHAR(5) NOT NULL  
);  
  
CREATE TABLE IF NOT EXISTS peminjaman (  
  kode_peminjaman VARCHAR (10) NOT NULL PRIMARY KEY,  
  id_anggota VARCHAR (10) NOT NULL,  
  id_petugas VARCHAR (10) NOT NULL,  
  tanggal_pinjam TIMESTAMP DEFAULT NOW(),  
  tanggal_kembali TIMESTAMP DEFAULT NOW(),  
  kode_buku VARCHAR (10) NOT NULL,  
  FOREIGN KEY (id_anggota) REFERENCES anggota(id_anggota),  
  FOREIGN KEY (id_petugas) REFERENCES petugas(id_petugas),  
  FOREIGN KEY (kode_buku) REFERENCES buku (kode_buku)  
);  
  
CREATE TABLE IF NOT EXISTS pengembalian (  
  kode_kembali VARCHAR (10) NOT NULL PRIMARY KEY,  
  id_anggota VARCHAR (10) NOT NULL,  
  kode_buku VARCHAR (10) NOT NULL,  
  id_petugas VARCHAR (10) NOT NULL,  
  tgl_pinjam TIMESTAMP DEFAULT NOW(),  
  tgl_kembali TIMESTAMP DEFAULT NOW(),  
  denda VARCHAR (15) NOT NULL,  
  FOREIGN KEY (id_anggota) REFERENCES anggota(id_anggota),  
  FOREIGN KEY (kode_buku) REFERENCES buku(kode_buku),  
  FOREIGN KEY (id_petugas) REFERENCES petugas(id_petugas)  
);  
  
SHOW TABLES;  
  
INSERT INTO anggota VALUES  
( 'A001','Tomo','Jakarta','2004-06-22',085211112222,'Laki-Laki'),  
( 'A002','Siska','Surabaya','2002-03-15',085233334444,'Perempuan'),  
( 'A003','Budi','Bandung','2001-11-08',085255556666,'Laki-Laki'),  
( 'A004','Dewi','Yogyakarta','2002-09-30',085277778888,'Perempuan'),  
( 'A005','Hadi','Gresik','2003-07-12',085299990000,'Laki-Laki'),  
( 'A006','Rina','Lamongan','2005-05-24',085211112233,'Perempuan'),  
( 'A007','Adi','Bandung','2001-01-18',085211112244,'Laki-Laki'),  
( 'A008','Lina','Jombang','2000-08-03',085211112255,'Perempuan'),  
( 'A009','Andi','Bojonegoro','2004-04-06',085211112266,'Laki-Laki'),  
( 'A010','Rani','Sidoarjo','2004-12-29',085211112277,'Perempuan');  
SELECT * FROM anggota;  
  
INSERT INTO buku VALUES  
( 'B001','Pulang','Tereliye','2017','9'),  
( 'B002','Matahari','Tereliye','2019','8'),  
( 'B003','Cahaya','Tereliye','2021','7');
```

4.2 Hasil

<input type="checkbox"/>	IdPetugas	Nama_Petugas	Jumlah_Peminjaman	Jumlah_Pengembalian	Jumlah_Buku_Terpinjam
<input type="checkbox"/>	1	John Doe	3	1	1
<input type="checkbox"/>	2	Jane Smith	3	1	1
<input type="checkbox"/>	3	Michael Johnson	3	1	1
<input type="checkbox"/>	4	Emily Brown	2	0	1
<input type="checkbox"/>	5	David Wilson	2	0	1
*	0	(NULL)	0	0	0

4.3 Penjelasan

- Pembuatan Tabel:

Tabel buku: Tabel ini berisi informasi tentang buku-buku yang tersedia di perpustakaan, seperti judul buku, pengarang, penerbit, tahun terbit, jumlah buku, status, dan klasifikasi.

Tabel petugas: Tabel ini mencatat informasi tentang petugas perpustakaan, termasuk ID petugas, username, password, dan nama petugas.

Tabel anggota: Tabel ini berisi informasi tentang anggota perpustakaan, seperti ID anggota, nama anggota, tahun anggota bergabung, tempat dan tanggal lahir, nomor telepon, jenis kelamin, dan status peminjaman.

Tabel peminjaman: Tabel ini mencatat informasi tentang peminjaman buku, termasuk ID peminjaman, ID anggota yang meminjam, ID petugas yang melakukan peminjaman, tanggal peminjaman, tanggal pengembalian, dan kode buku yang dipinjam.

Tabel pengembalian: Tabel ini mencatat informasi tentang pengembalian buku, termasuk ID anggota yang mengembalikan, kode buku yang dikembalikan, ID petugas yang menerima pengembalian, tanggal peminjaman, tanggal pengembalian, dan denda (jika ada).

- Penambahan Data:

Data ditambahkan ke tabel buku, petugas, anggota, peminjaman, dan pengembalian menggunakan perintah INSERT INTO.

Pembuatan View:

Anggota_Lebih_5_Buku: View ini menampilkan anggota perpustakaan yang telah melakukan lebih dari 5 transaksi peminjaman.

Transaksi_Petugas: View ini menampilkan statistik transaksi untuk setiap petugas, termasuk jumlah peminjaman yang dilakukan, jumlah pengembalian yang diterima, dan jumlah buku yang terpinjam.

Petugas_Teratas: View ini menampilkan petugas perpustakaan dengan jumlah total transaksi (peminjaman dan pengembalian) tertinggi.

Buku_Terpinjam_Terbanyak: View ini menampilkan buku yang paling sering dipinjam dari perpustakaan, berdasarkan jumlah peminjaman.

BAB V

PENUTUP

5.1 Analisa

Dari hasil praktikum, praktikan menganalisa bahwa penjelasan mengenai modul View, kami telah menjalankan serangkaian tugas yang bertujuan untuk memperdalam pemahaman kami tentang konsep-konsep dasar dalam pengelolaan basis data menggunakan view. View merupakan objek yang berfungsi untuk menampilkan data dari satu atau lebih tabel dalam basis data, dan memungkinkan untuk mengatur tampilan data sesuai kebutuhan pengguna tanpa mengubah struktur tabel yang mendasarinya.

Pada awal praktikum, kami mempelajari konsep dasar tentang bagaimana membuat, mengubah, dan menghapus view dalam sistem manajemen basis data yang digunakan. Proses ini melibatkan penulisan perintah SQL yang tepat untuk mendefinisikan struktur view serta klausul-klausul yang dapat digunakan untuk memodifikasi sifat-sifat dari view tersebut.

Selanjutnya, kami melakukan eksperimen dengan penggunaan view untuk menyederhanakan akses data dan mengurangi kompleksitas query yang diperlukan. Dengan memanfaatkan view, kami dapat menyembunyikan rincian kompleks dari struktur tabel yang mendasari dan menyediakan antarmuka yang lebih intuitif untuk mengakses data. Selama praktikum, kami juga mempelajari manfaat dari penggunaan view dalam konteks keamanan data. Dengan memberikan akses terhadap view yang telah ditentukan, kami dapat mengontrol dengan lebih tepat siapa saja yang memiliki hak akses terhadap data tertentu dalam basis data.

5.2 Kesimpulan

1. Pengurangan Kompleksitas Query: Penggunaan view memungkinkan pengurangan kompleksitas query dengan menyembunyikan rincian struktur tabel yang mendasari, sehingga meningkatkan efisiensi dalam akses data.
 2. View dapat digunakan untuk mengontrol akses data dengan lebih tepat, memberikan lapisan keamanan tambahan terhadap informasi sensitif dalam basis data.
-

LAPORAN RESMI
MODUL III
STORED PROCEDURE
SISTEM MANAJEMEN BASIS DATA



NAMA	: FAIRUZ ABDULLAH
N.R.P	: 220441100070
DOSEN	: MOHAMMAD SYARIEF, ST., MCs
ASISTEN	: NURI HIDAYATULOH
TGL PRAKTIKUM	: 26 APRIL 2024

Disetujui : 02 Maret 2024
Asisten

NURI HIDAYATULOH
210441100100



LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I

PENDAHULUAN

1.1 Latar Belakang

Stored Procedure dalam pengembangan basis data modern sangat penting dalam konteks efisiensi, konsistensi, dan keamanan data. Dalam era aplikasi berbasis data yang semakin kompleks dan tuntutan terhadap kinerja yang tinggi, Stored Procedure menjadi salah satu solusi utama untuk menangani skenario yang memerlukan logika bisnis kompleks di dalam database. Dengan menyimpan logika bisnis di dalam Stored Procedure, pengembang dapat mengurangi jumlah permintaan jaringan antara aplikasi dan database, menghasilkan latensi yang lebih rendah dan kinerja yang lebih baik secara keseluruhan.

Selain itu, Stored Procedure juga memainkan peran penting dalam menjaga keamanan dan integritas data. Dengan mengontrol akses ke database melalui prosedur yang telah ditentukan, pengelola database dapat memastikan bahwa hanya pengguna yang berwenang yang dapat melakukan operasi tertentu, mencegah potensi ancaman keamanan seperti injeksi SQL dan akses yang tidak sah. Dalam hal ini, Stored Procedure berfungsi sebagai lapisan keamanan tambahan yang dapat memberikan perlindungan lebih lanjut terhadap data sensitif dan kritis dalam sistem basis data. Dengan demikian, pemahaman yang baik tentang konsep dan penggunaan Stored Procedure akan membantu pengembang membangun aplikasi yang efisien, aman, dan dapat diandalkan di lingkungan bisnis yang semakin kompleks.

1.2 Tujuan

- Mampu memahami dan membuat procedure dalam basis data dan mampu menggunakan perintah-perintah dalam stored procedure serta menyelesaikan operasi – operasi data spesifik dengan memanfaatkan stored procedure.
-

BAB II DASAR TEORI

2.1 Dasar Teori

Stored Procedure adalah sebuah prosedur layaknya subprogram (subrutin) di dalam bahasa pemrograman reguler yang tersimpan di dalam katalog basis data. Beberapa kelebihan yang ditawarkan stored procedure antara lain : meningkatkan performa, mereduksi trafik jaringan, reusable, dan meningkatkan kontrol sekuriti. Di balik kelebihan tersebut, stored procedure juga memiliki kekurangan.

sintaks stored procedure :

```
<create procedure statement> ::=
CREATE PROCEDURE <procedure name> ( [ <parameter
list>
] )
<routine body>
<parameter list> ::=
<parameter specification> [ ,
<parameterspecification> ]...
<parameter specification> ::=
[ IN | OUT | INOUT ] <parameter> <data type>
<routine body> ::= <begin-end block>
<begin-end block> ::=
[ <label> : ] BEGIN <statement list> END [ <label>
]
<statement list> ::= { <body statement> ; }...
<statement in body> ::=
```

pernyataan pembuatan stored procedure berikut :

```
DELIMITER //
CREATE PROCEDURE getMhs()
BEGIN
    SELECT * FROM mahasiswa;
END //
DELIMITER;
```

```
<call statement> ::=  
CALL [ <database name> . ] <stored procedure name>  
( [ <scalar expression> [ , <scalar expression> ]... ] )
```

Aktivasi/pemanggilan Stored Procedure:

Eksekusi Query tersebut dengan memanggil procedure getMahasiswa().

```
CALL getMhs();
```

```
<drop procedure statement>  
::=DROP PROCEDURE [ IF  
EXISTS ]  
[ <database name> . ] <procedure name>
```

Menghapus Stored Procedure:

Dalam Implementasinya, penggunaan stored procedure sering melibatkan parameter. Di MySQL, parameter stored procedure dibedakan menjadi tiga mode : IN, OUT, dan INOUT.

2.2 IN

Parameter yang merupakan mode default ini mengindikasikan bahwa sebuah parameter dapat di-pass ke dalam stored procedure tetapi nilainya tidak dapat diubah (dari dalam stored procedure

Sebagai contoh, kita bisa mendapatkan semua data matakuliah di semester tertentu.

```
DELIMITER //  
CREATE PROCEDURE getNamaByJenisKlamin(IN jeniskelamin varchar(1))  
BEGIN  
    select * from mhs  
    where jKlmn = jeniskelamin;  
END //  
DELIMITER ;
```

Untuk memanggil stored procedure yang memiliki parameter, maka kita harus menspesifikasikan argumennya. Misalkan kita ingin mendapatkan data mahasiswa dengan jenis kelamin laki-laki.

CALL getNamaByJenisKlamin(3)

idMhs	namaMhs	almtMhs	jKlmn
1011	Ari	Surabaya	L
1012	Bagus	Malang	L
1013	Candra	Sampang	L

Apabila pemanggilan stored procedure di atas mengabaikan argumen, DBMS akan merespon dengan pesan kesalahan. Bergantung kebutuhan, pendefinisian parameter pada stored procedure juga bisa lebih dari satu. Sebagai contoh, buat stored procedure dengan dua buah parameter

```
DELIMITER //
CREATE PROCEDURE getNamaByAlmtJKelamin(
    IN Alamat VARCHAR(20),
    IN jenisKelamin VARCHAR(2))
BEGIN
    SELECT * FROM mhs
    WHERE jKlmn = jenisKelamin
    AND almtMhs = Alamat ;
END //
DELIMITER ;
```

seperti berikut :

Pemanggilan stored procedure di atas tentunya akan memerlukan dua buah argumen.

CALL getNamaByAlmtJKelamin ('surabaya', 'L');

```
DELIMITER //
CREATE PROCEDURE AddMahasiswa(
    IN idMhs INT(10),
    IN namaMhs VARCHAR(20),
    IN almtMhs VARCHAR(30),
    IN jKlmn VARCHAR(2))
BEGIN
    INSERT INTO mhs
    VALUES (idMhs, namaMhs, almtMhs, jKlmn);
END //
DELIMITER ;
```

Penambahan Data Pada operasi penambahan, data – data terkait diisikan melalui argumen. Selanjutnya, isi stored procedure akan memasukkan data ke dalam tabel. Berikut adalah contoh

stored procedure untuk menambahkan data pada tabel mahasiswa.

Lakukan eksekusi terhadap procedure tersebut

call AddMahasiswa('1019','Gunawan','surabaya', 'L');

Selanjutnya lakukan pengecekan data pada tabel mahasiswa.

select * from mhs;

atau select *

idMhs	namaMhs	almtMhs	jKlmn
1011	Ari	Surabaya	L
1012	Bagus	Malang	L
1013	Candra	Sampang	L
1014	Dhani	Surabaya	P
1018	Frinza	Kediri	P
1019	Gunawan	surabaya	L

getMhs();

2.3 OUT

Mode ini mengindikasikan bahwa stored procedure dapat mengubah parameter dan mengirimkan kembali ke program pemanggil. Dalam konteks bahasa pemrograman, parameter OUT analog dengan passing-byreference.

```
DELIMITER //
CREATE PROCEDURE JumlahMahasiswa(OUT jumlah_mhs INT(3))
BEGIN
    SELECT COUNT(idMhs)
    INTO jumlah_mhs FROM mhs;
END //
DELIMITER ;
```

Dengan demikian, parameter ini nilainya bisa diubah oleh stored procedure.

Untuk mengeksekusi stored procedure dengan parameter OUT, dibutuhkan argumen yang spesifik.

call JumlahMahasiswa(@jumlah_mhs);

Perhatikan, argumen harus menggunakan notasi @, yang mengindikasikan sebagai suatu parameter OUT.

Langkah selanjutnya, untuk mendapatkan nilai variabel, gunakan pernyataan SELECT

select @jumlah_mhs;

@jumlah_mhs

Parameter mode OUT juga bisa dikombinasikan dengan mode IN.

2.4 INOUT

Mode ini pada dasarnya merupakan kombinasi dari mode IN dan OUT. Kita bisa mengirimkan parameter ke dalam stored procedure dan mendapatkan nilai kembalian yang baru dari stored procedure yang didefinisikan.

```
DELIMITER //
CREATE PROCEDURE CountByGender(
    IN gender VARCHAR(2),
    OUT total INT(3))
BEGIN
    SELECT COUNT(idMhs)
    INTO total
    FROM mhs
    WHERE jKlmn = gender;
END //
DELIMITER ;
```

Sebagai contoh, definisikan stored procedure seperti berikut :

Lakukan eksekusi pada procedure tersebut untuk mencari jumlah mahasiswa yang berjenis kelamin perempuan .

Stored procedure dapat mencerminkan beragam operasi data, misalnya seleksi, penambahan, pengubahan, penghapusan, dan juga operasi – oprasi DDL. Seperti

```
call CountByGender('P',@total);
select @total;
```

@total

2

halnya procedure di dalam bahasa pemrograman, stored procedure juga dapat melibatkan variabel, pernyataan kondisional, dan pengulangan.

Alhas
11

BAB II

TUGAS PENDAHULUAN

3.1 Soal

1. Apa Perbedaan antara Stored Procedure dan fungsi dalam SQL
2. Apa keuntungan menggunakan Stored Procedure dalam aplikasi database?
3. Apa itu EXECUTE dalam SQL dan bagaimana cara menggunakannya untuk menjalankan Stored Procedure
4. Apa saja jenis-jenis Stored Procedure dalam SQL?

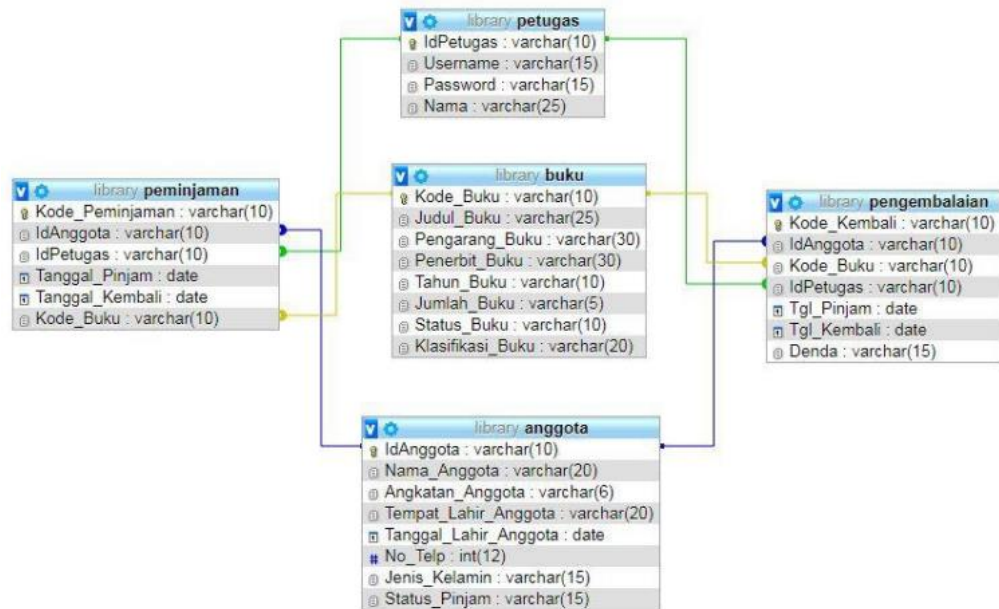
3.2 Jawaban

1. Stored Procedure : kumpulan pernyataan SQL yang telah disimpan didalam basis data dan dapat dieksekusi secara berulang tanpa menulis ulang kodennya
Fungsi : merupakan kumpulan pernyataan SQL juga namun selalu mengembalikan nilai.
2. - kinerja yang lebih baik
 - keamanan
 - kemudahan pemeliharaan
 - konsistensi data
3. Execute adalah Pernyataan SQL untuk mengeksekusi stored procedure, cara : EXECUTE nama-prosedur, nilai;
4. - Tidak berparameter
 - dengan parameter satu atau lebih
 - DML untuk memanipulasi data
 - DDL untuk mengelompokkan struktur database

BAB IV

IMPLEMENTASI

4.1 Soal



1. Buatlah sebuah prosedur dengan menggunakan parameter IN default
2. Definisikan stored procedure untuk mengetahui data pada salah satu table dengan berdasarkan pada salah satu atribut/field
3. Definisikan stored procedure untuk mengetahui data pada salah satu table dengan berdasarkan pada dua atribut/field.
4. Definisikan stored procedure untuk mengetahui data pada salah satu table dengan berdasarkan pada tiga atribut/field
5. Definisikan stored procedure untuk memasukkan data pada salah satu table
6. Definisikan stored procedure untuk mengetahui data pada salah satu table menggunakan parameter OUT
7. Definisikan stored procedure untuk mengetahui data pada salah satu table menggunakan parameter INOUT

4.2 Source Code

```
CREATE DATABASE IF NOT EXISTS modul3;
USE modul3;
CREATE TABLE IF NOT EXISTS anggota (
id_anggota VARCHAR(10) NOT NULL PRIMARY KEY,
nama_anggota VARCHAR(20) NOT NULL,
angkatan_anggota VARCHAR(6) NOT NULL,
tempat_lahir_anggota VARCHAR(20) NOT NULL,
tanggal_lahir_anggota DATE,
no_telp INT(12) NOT NULL,
jenis_kelamin VARCHAR(15) NOT NULL,
status_pinjam VARCHAR(15) NOT NULL
);
CREATE TABLE IF NOT EXISTS petugas (
id_petugas VARCHAR(10) NOT NULL PRIMARY KEY,
username VARCHAR(15) NOT NULL,
PASSWORD VARCHAR(15) NOT NULL,
nama VARCHAR(25) NOT NULL
);
CREATE TABLE IF NOT EXISTS buku (
kode_buku VARCHAR(10) NOT NULL PRIMARY KEY,
judul_buku VARCHAR(25) NOT NULL,
pengarang_buku VARCHAR(30) NOT NULL,
tahun_buku VARCHAR(10) NOT NULL,
jumlah_buku VARCHAR(5) NOT NULL,
status_buku VARCHAR(10) NOT NULL,
klasifikasi_buku VARCHAR(20) NOT NULL
);
CREATE TABLE IF NOT EXISTS peminjaman (
kode_peminjaman VARCHAR(10) NOT NULL PRIMARY KEY,
id_anggota VARCHAR(10) NOT NULL,
id_petugas VARCHAR(10) NOT NULL,
tanggal_pinjam DATE,
tanggal_kembali DATE,
kode_buku VARCHAR(10) NOT NULL,
FOREIGN KEY (id_anggota) REFERENCES anggota(id_anggota),
FOREIGN KEY (id_petugas) REFERENCES petugas(id_petugas),
FOREIGN KEY (kode_buku) REFERENCES buku(kode_buku)
);
CREATE TABLE IF NOT EXISTS pengembalian (
kode_kembali VARCHAR(10) NOT NULL PRIMARY KEY,
id_anggota VARCHAR(10) NOT NULL,
kode_buku VARCHAR(10) NOT NULL,
id_petugas VARCHAR(10) NOT NULL,
tgl_pinjam DATE,
tgl_kembali DATE,
denda VARCHAR(15) NOT NULL,
FOREIGN KEY (id_anggota) REFERENCES anggota(id_anggota),
FOREIGN KEY (kode_buku) REFERENCES buku(kode_buku),
FOREIGN KEY (id_petugas) REFERENCES petugas(id_petugas)
);
SHOW TABLES;
INSERT INTO anggota VALUES
('A001','Tomo','23','Jakarta','2004-06-22','085211112222','Laki-Laki','Pinjam'),
('A002','Siska','20','Surabaya','2002-03-15','085233334444','Perempuan','Pinjam'),
('A003','Budi','20','Bandung','2001-11-08','085255556666','Laki-Laki','Kembali'),
('A004','Dewi','21','Yogyakarta','2002-09-30','085277778888','Perempuan','Pinjam'),
('A005','Hadi','22','Gresik','2003-07-12','085299990000','Laki-Laki','Kembali'),
('A006','Rina','24','Lamongan','2005-05-24','085211112233','Perempuan','Pinjam'),
('A007','Adi','19','Bandung','2001-01-18','085211112244','Laki-Laki','Pinjam'),
('A008','Lina','18','Jombang','2000-08-03','085211112255','Perempuan','Kembali'),
('A009','Andi','22','Bojonegoro','2004-04-06','085211112266','Laki-Laki','Pinjam'),
('A010','Rani','23','Sidoarjo','2004-12-29','085211112277','Perempuan','Kembali');
SELECT * FROM anggota;
INSERT INTO buku VALUES
('B001','Pulang','Tereliye','2017','9','Tersedia','Fiksi'),
('B002','Matahari','Tereliye','2019','8','Tersedia','Fantasi'),
('B003','Senja','Biru','2018','7','Tersedia','Romantis'),
('B004','Pelangi','Jompi','2020','6','Tersedia','Anak-anak'),
('B005','Cahaya','Sukarso','2016','5','Tersedia','Pendidikan'),
('B006','Bulan','Terelive','2015','4','Tersedia','Fiksi'),
```

4.3 Hasil

4.3.1 Nomor 1

kode_peminjaman	id_anggota	id_petugas	tanggal_pinjam	tanggal_kembali	kode_buku
PM009	A010	P002	2024-04-30 10:16:40	2024-05-07 10:16:40	B005
PM010	A007	P003	2024-04-30 10:16:40	2024-05-07 10:16:40	B002
PM011	A005	P005	2024-04-30 10:16:40	2024-05-07 10:16:40	B008
PM012	A005	P005	2024-04-30 10:16:40	2024-05-07 10:16:40	B003
PM013	A005	P003	2024-04-30 10:16:40	2024-05-07 10:16:40	B001
PM014	A005	P003	2024-04-30 10:16:40	2024-05-07 10:16:40	B009
PM015	A001	P005	2024-04-30 10:29:24	2024-04-30 10:29:24	B009
PM025	A001	P001	2024-04-30 11:10:14	2024-04-30 11:10:14	B001

4.3.2 Nomor 2

id_anggota	nama_anggota	tempat_lahir_anggota	tanggal_lahir_anggota	no_telp	jenis_kelamin
A001	Tomo	Jakarta	2004-06-22	2147483647	Laki-Laki

4.3.3 Nomor 3

kode_buku	judul_buku	pengarang_buku	tahun_buku	jumlah_buku
B001	Pulang	Tereliye	2017	9

id_petugas	username	PASSWORD	nama
P008	Anto90	An08	Anton

4.3.4 Nomor 4

4.3.5 Nomor 5

<input type="checkbox"/>	PM012	A005	P005	2024-04-30 10:16:40	2024-05-07 10:16:40	B003
<input type="checkbox"/>	PM013	A005	P003	2024-04-30 10:16:40	2024-05-07 10:16:40	B001
<input type="checkbox"/>	PM014	A005	P003	2024-04-30 10:16:40	2024-05-07 10:16:40	B009
<input type="checkbox"/>	PM015	A001	P005	2024-04-30 10:29:24	2024-04-30 10:29:24	B009
<input type="checkbox"/>	PM025	A001	P001	2024-04-30 11:10:14	2024-04-30 11:10:14	B001

4.3.6 Nomor 6

@jumlah_buku
17

4.3.7 Nomor 7

@jumlah_buku
7

4.4 Penjelasan

4.4.1 Nomor 1

Prosedur ini digunakan untuk memasukkan data peminjaman baru ke dalam tabel `peminjaman`. Dengan memberikan nomor peminjaman, ID anggota, ID petugas, dan

kode buku, prosedur ini memasukkan data peminjaman baru ke dalam database. Proses ini digunakan untuk memudahkan dan memastikan konsistensi saat memasukkan data peminjaman baru ke dalam database. Dengan menggunakan prosedur tersimpan, kita dapat menentukan secara konsisten format dan parameter yang diperlukan untuk memasukkan data baru, mengurangi kesalahan dan kompleksitas di sisi pengguna aplikasi.

4.4.2 Nomor 2

Prosedur ini digunakan untuk mencari anggota berdasarkan nama anggota yang diberikan sebagai parameter. Prosedur ini memberikan fleksibilitas dalam melakukan pencarian data dalam database. Dengan menyediakan nama anggota, pengguna dapat dengan mudah menemukan informasi yang mereka butuhkan tanpa perlu mengetahui detail teknis bagaimana pencarian dilakukan di dalam database.

4.4.3 Nomor 3

Prosedur ini digunakan untuk mencari buku berdasarkan kode buku dan nama pengarang yang diberikan sebagai parameter. Sama seperti prosedur sebelumnya, ini memberikan cara yang mudah untuk mencari data buku dengan memberikan informasi yang dimiliki pengguna (kode buku dan nama pengarang). Ini meningkatkan keterbacaan dan keterjangkauan kode.

4.4.4 Nomor 4

Prosedur ini digunakan untuk mencari petugas berdasarkan ID petugas, nama pengguna, dan nama petugas yang diberikan sebagai parameter. Proses ini sering kali diperlukan dalam aplikasi manajemen untuk memverifikasi identitas petugas. Dengan menggunakan prosedur ini, pencarian dilakukan dengan konsisten dan efisien, tanpa perlu menuliskan kueri pencarian secara langsung setiap kali diperlukan.

4.4.5 Nomor 5

Prosedur ini mirip dengan `InDefault`, digunakan untuk memasukkan data peminjaman baru ke dalam tabel `peminjaman`, namun menggunakan nama yang berbeda. Penggunaan prosedur tersimpan di sini mengikuti pola yang sama dengan `InDefault`, namun dengan nama yang berbeda. Ini bisa saja dilakukan untuk alasan organisasi kode atau kejelasan logika program.

4.4.6 Nomor 6

Prosedur ini digunakan untuk menghitung jumlah total baris dalam tabel `peminjaman` dan menyimpannya dalam variabel `jumlah_buku` sebagai output. Proses ini menunjukkan bagaimana kita dapat menggunakan variabel output untuk mengembalikan hasil dari prosedur tersimpan. Dengan cara ini, kita dapat mengembalikan nilai yang dihasilkan oleh prosedur untuk digunakan di luar prosedur tersebut.

4.4.7 Nomor 7

Prosedur ini digunakan untuk menambah stok buku berdasarkan kode buku yang diberikan sebagai parameter. Proses ini sering kali diperlukan dalam manajemen stok barang.

BAB V PENUTUP

5.1 Analisa

Dari hasil praktikum, praktikan menganalisa bahwa penggunaan Stored Procedure memberikan beberapa manfaat yang signifikan dalam pengembangan aplikasi basis data. Pertama, penggunaan Stored Procedure meningkatkan efisiensi kinerja aplikasi dengan mengurangi jumlah permintaan jaringan antara aplikasi dan database. Dengan logika bisnis yang terkonsolidasi di dalam Stored Procedure, aplikasi dapat mengakses data dengan cepat dan efisien, menghasilkan waktu respons yang lebih singkat dan meningkatkan pengalaman pengguna secara keseluruhan.

Selain itu, praktikan juga menyimpulkan bahwa penggunaan Stored Procedure membantu meningkatkan keamanan data dengan menyediakan lapisan keamanan tambahan di dalam database. Dengan menyimpan logika bisnis di dalam Stored Procedure, pengembang dapat mengontrol akses ke data dengan lebih baik dan mencegah potensi ancaman keamanan seperti injeksi SQL. Hal ini memberikan perlindungan yang lebih kuat terhadap data sensitif dan kritis dalam sistem basis data, mengurangi risiko pelanggaran keamanan dan kebocoran informasi.

5.2 Kesimpulan

1. Penggunaan Stored Procedure secara signifikan meningkatkan efisiensi kinerja aplikasi dengan mengurangi jumlah permintaan jaringan antara aplikasi dan database.
 2. Stored Procedure memberikan lapisan keamanan tambahan di dalam database dengan menyimpan logika bisnis di dalamnya.
 3. Dengan menyediakan satu titik masuk untuk memanipulasi data, Stored Procedure membantu memastikan konsistensi dan integritas data dalam sistem basis data.
-

LAPORAN RESMI
MODUL V DAN VI
SQL DAN TYPE DATA/ STORED PROCEDURE
SISTEM MANAJEMEN BASIS DATA



NAMA	: FAIRUZ ABDULLAH
N.R.P	: 220441100070
DOSEN	: MOHAMMAD SYARIEF, ST., M.Cs
ASISTEN	: NURI HIDAYATULOH
TGL PRAKTIKUM	: 17 MEI 2024

Disetujui 09 Mei 2024
Asisten

NURI HIDAYATULOH
210441100100



LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I

PENDAHULUAN

1.1 Latar Belakang

Sistem Manajemen Basis Data (SMBD) merupakan fondasi utama dalam pengelolaan data yang efektif dan efisien dalam sebuah organisasi atau perusahaan. Dalam rangka meningkatkan pemahaman dan keterampilan praktis mahasiswa dalam mengelola data menggunakan SMBD, dilakukanlah praktikum dengan materi fokus pada bahasa kueri SQL (Structured Query Language).

SQL adalah bahasa standar yang digunakan untuk mengakses dan mengelola basis data relasional. Dengan SQL, pengguna dapat melakukan berbagai operasi, mulai dari menyimpan, mengambil, memperbarui, hingga menghapus data dalam basis data. Praktikum ini bertujuan untuk memberikan pemahaman mendalam tentang penggunaan SQL dalam mengelola basis data, serta penerapan berbagai perintah dan teknik seperti Select Statement, Menyaring Data dengan klausa WHERE, Pengurutan Data dengan ORDER BY, Operator BETWEEN untuk memfilter rentang nilai, Operator IN untuk nilai dalam daftar, Operator LIKE untuk mencocokkan pola string, Ekspresi Kueri, Agregasi dan Pengelompokan Data dengan fungsi, Multiple-table Query dengan JOIN, dan penggunaan Subquery.

Dengan memahami dan menguasai berbagai perintah dan teknik SQL tersebut, diharapkan mahasiswa dapat lebih siap dalam mengelola dan menganalisis data menggunakan SMBD dalam konteks dunia nyata. Praktikum ini juga membantu mahasiswa untuk mengembangkan keterampilan dalam merancang kueri yang efisien dan efektif untuk memenuhi kebutuhan analisis data yang beragam.

1.2 Tujuan

- Mampu memahami dan membuat perintah SQL lebih dalam dengan berbagai kondisi, Ekspresi, pengurutan dan mampu mengabungkan beberapa argument / perintah pada beberapa tabel (join) dalam basis data.
 - Memahami dan membuat Stored Procedure pada basisdata.
 - Mampu Menggunakan type data sesuai kebutuhan pada Stored Procedure.
-

BAB II

DASAR TEORI

2.1 SQL

Secara umum perintah-perintah yang terdapat di dalam SQL, diklasifikasikan menjadi tiga bagian, antara lain yaitu:

- a. DDL (Data Definition Language)
 - Merupakan perintah SQL yang berkaitan dengan pendefinisian suatu struktur database, dalam hal ini database dan table.
 - Perintah DDL adalah: CREATE, ALTER, RENAME, DROP
- b. DML (Data Manipulation Language)
 - Merupakan perintah SQL yang berkaitan dengan manipulasi atau pengolahan data atau record dalam table.
 - Perintah DML antara lain: SELECT, INSERT, UPDATE, DELETE
- c. DCL (Data Control Language)
 - Merupakan perintah SQL yang berkaitan dengan manipulasi user dan hak akses (priviledges).
 - Perintah SQL yang termasuk dalam DCL antara lain: GRANT, REVOKE.

2.2 SELECT

SELECT merupakan salah satu pondasi dalam SQL Programming. SELECT digunakan untuk menampilkan data, terlebih untuk mencari informasi dalam kumpulan data.

a. Sintak

SELECT dibagi kedalam 6 komponen, antara lain:

- SELECT. Diikuti oleh , dapat berupa literal_value atau column_list atau asterisk (*).
 - FROM. Diikuti oleh sesuai dengan column_list. Jadi jika ada data yang diambil dari kolom tertentu, harus diketahui kolom tersebut diambil dari tabel mana. Tabel pada FROM dapat diikuti dengan alias untuk mempermudah penulisan khususnya ketika join dan subquery.
 - WHERE. Diikuti oleh kondisi secara umum.
-

- GROUP BY. Diikuti oleh . Bagian ini muncul ketika ada fungsi fungsi agregasi.
- HAVING. Diikuti oleh kondisi hanya untuk fungsi -fungsi agregasi.
- ORDER BY. Diikuti oleh <select_list>, perintahnya;

```
SELECT      <select_list>
[FROM      <table_name>]
[WHERE      <kondisi1> [AND/OR <kondisi2>]]
[GROUP BY  <select_list>]
[HAVING     <kondisi1> [AND/OR <kondisi2>]]
[ORDER BY  <select_list>]
```

Keterangan :

1. SELECT, INTO, FROM, WHERE, GROUP BY, HAVING DAN ORDER BY → kata kunci (keyword) yang harus disertakan jika kita membutuhkannya di dalam pengolahan data.
2. select_list, table_source, search_condition, group_by_expression, order_expression → isian yang bisa kita ubah berdasarkan kebutuhan kita
3. Kurung kotak [] → bagian tersebut boleh disertakan atau tidak, tergantung dari kebutuhan

2.3 Menyaring Data

Tidak semua data yang ada pada tabel, ingin ditampilkan. Terlebih ketika tabel terbagi kedalam banyak kolom dengan jumlah data yang sangat besar. Operator Pembandingan:

Operator	Keterangan
=	Sama dengan
>	Lebih besar dari
>=	Lebih besar sama dengan
<	Kurang dari
<=	Kurang dari sama dengan
<> atau !=	Tidak sama dengan
BETWEEN .. AND ..	Diantara 2 nilai
IN (set)	Cocok dengan salah satu diantara daftar nilai
LIKE	Cocok dengan pola karakter
IS NULL	Sama dengan NULL

- Perintahnya:

Select * From Nama_Table Where Nama_Field [Operator Relasional]

Ketentuan;

- Contoh :

```
select * from customer where cust_id = '10003' or cust_name = 'wascals';
```

2.4 Pengurutan data (ASC, DESC, ORDER BY)

Untuk mengurutkan tampilan data dari suatu table, digunakan klausa Order By. Klausa Order By, dapat digunakan untuk mengurutkan data;

- a. Asc (Ascending) : Untuk mengurutkan data dari kecil ke besar
- b. Desc (Descending) : Untuk mengurutkan data dari besar ke kecil
Perintahnya `Select * From Nama_Table Order By Nama_Field_Key Asc/Desc;`

Contoh : `Select * From products Order By prod_name Asc;`

2.5 OPERATOR BETWEEN, IN, LIKE

- a. Operator Between Operator Between merupakan operator yang digunakan untuk menangani operasi jangkauan.

- Perintahnya:

```
Select * From Nama_Table Where Nama_Field_ketentuan Between 'Ketentuan_1' And 'Ketentuan_2';
```

- Contoh:

```
Select * From orderitems Where quantity Between '1' And '10';
```

- b. Operator In Operator In merupakan operator yang digunakan untuk mencocokkan suatu nilai.

- Perintahnya:

```
Select Nama_Field From Nama_Table Where Nama_Field_Pencocok In ('Isi_Field_1','Isi_Field_2');
```

- Contoh:

Menampilkan nama customer, alamat dan email customer tertentu.

```
Select cust_name,cust_address,cust_email From customers Where cust_id In ('10002','10005');
```

- c. Operator Like Operator Like merupakan operator yang digunakan untuk mencari suatu data (search).
-

- Perintahnya :
Select * From Nama_Table Where Nama_Field_Dicari Like '%Key';
- Contoh :
Select * From Products Where prod_name Like '%s'; Query yang pertama menampilkan produk dengan nama produk diawali huruf dan pada query yang kedua nama produk diakhiri huruf s.

2.6 Ekspresi Query

Ekspresi Query dapat digunakan untuk melakukan perubahan terhadap field kolom keluaran, menambah baris teks field keluaran.

a. Mengganti Nama Field

- keluaran Perintahnya:
Select Nama_Field_Asal As 'Nama_Field_Pengganti' From Nama_Table;
- Contoh:
Select Kode_Mtkul As 'Kode Matakuliah', Nama_Mtkul As 'Matakuliah' From Mtkul;

b. Menambahkan Baris Teks Field Keluaran

- Perintahnya:
Select 'Nama Field Tambahan', Nama_Field_Asal From Nama_Table;
- Contoh:
Select vend_name, 'diproduksi di', vend_city From vendors;

c. Ekspresi Kondisi

- Perintahnya:
Select Nama_Field_1 Case Nama_Field_2 When 'Nilai_field_2' Then 'Keterangan_1' Else 'Keterangan_2' End As Nilai_field_2 From Nama_Table;
- Contoh:
Select Kode_Mtkul, Nama_Mtkul, Case Sks When '1' Then 'Praktikum' Else 'Matakuliah' End As Sks From Mtkul;

2.7 Agregasi dan Pengelompokan Data

Dalam pemrosesan data mentah menjadi data statistik, diperlukan fungsi-fungsi

yang dapat meng-agregasi data-data tersebut. Fungsi-fungsi ini meliputi SUM, MIN, MAX, COUNT, dan AVG.

Fungsi	Keterangan
AVG	Menghitung rata-rata
COUNT	Menghitung cacah data /jumlah baris
MAX	Memperoleh nilai terbesar
MIN	Memperoleh nilai terkecil
SUM	Memperoleh jumlahan data

2.8 Multiple-table Query

Data-data yang tersimpan dalam basis data, tersebar kedalam beberapa tabel. Tabel-tabel ini dihubungkan dengan yang namanya referential constraint, yaitu hubungan antara foreign key dan primary key. Karena itulah, untuk mendapatkan informasi yang tersebar, dibutuhkan metode untuk menggabungkan property tabel-tabel tersebut. Metode yang digunakan ada 2 macam, yaitu join dan subquery. Perbedaannya sederhana, join menggunakan satu SELECT, sedangkan subquery menggunakan dua atau lebih SELECT (umumnya dikatakan sebagai SELECT within a SELECT).

- Join

Bentuk join pertama kali adalah menggunakan kata kunci WHERE untuk melakukan penggabungan tabel

```
SELECT <select_list>
FROM   <table1>, <table2> [, ...]
WHERE  <table1.PK = table2.FK> [AND ...]
```

Perkembangan SQL ANSI

```
SELECT <select_list>
FROM   <table1> JOIN <table2>
      ON < table1.PK = table2.FK> [[AND ...]
      JOIN ...];
```

- Tipe Join

Ada 2 tipe join, yaitu inner join yang lebih menekankan pada keberadaan

data yang sama, dan outer join.

```
SELECT <select_list>
FROM   <tabel1 sebagai kiri>
       <LEFT/RIGHT> [OUTER] JOIN
       <tabel2 sebagai kanan>
       ON <table1.PK = table2.FK> [AND ...];
```

Pada INNER JOIN atau CROSS JOIN output/hasil yang ditampilkan adalah data data dari semua table yang terlibat dimana baris yang tampil hanya yang memiliki kondisi kesamaan data. Kesamaan data berdasarkan relasinya (kesamaan data foreign key dengan primary key tabel yang diacu).

Berikut adalah bentuk umum INNER JOIN yang umumnya hanya disebut sebagai JOIN: SELECT nm_tabel1.nm_kolom1, nm_tabel1.nm_kolom2, nm_tabel2.nm_kolom1, nm_tabel2.nm_kolom2 FROM tabel1, tabel2 WHERE tabel1.nama_kolom1 (primary key)=tabel2.nama_kolom1(foreign key yg mengacu ke tabel1)

Contoh penggunaan Join, kita lihat kembali skema order entry dibawah ini. Menampilkan prod_name, vend_name dari table vendors dan products. Select vendors.vend_name,products.prod_name from vendors, products Where vendors.vend_id = products.vend-id

a. Clausa Join On Alias

```
SELECT a.nm_kolom1, b.nm_kolom2, a.nm_kolom3 FROM tabel1 a
JOIN      tabel2      b      ON      a.nama_kolom1(primary
key)=b.nama_kolom1(foreign key yg mengacu ke tabel1) WHERE
kondisi;
```

b. JOIN 3 TABLE Atau Lebih

Pada prinsipnya sama, hanya jumlah tabel ditambah dan sintaks disesuaikan. Contoh penerapan join dua tabel atau lebih untuk menampilkan nama customer, tgl order dan total jumlah order.

```
select a.cust_name,b.order_date,c.quantity from customers a join orders b
on a.cust_id=b.cust_id join orderitems c on b.order_num=c.order_num;
```

c. OUTER JOIN

Pada OUTER JOIN hasil yang ditampilkan adalah data-data dari semua tabel yang terlibat baik yang hanya yang memiliki kondisi kesamaan data berdasarkan relasinya (kesamaan data foreign key dengan primary key tabel yang diacu) maupun data yang tidak memiliki kesamaan data berdasarkan relasinya dari salah satu tabel. Terdapat dua tipe OUTER JOIN, yaitu:

- LEFT OUTER JOIN atau biasa disebut left join
- RIGHT OUTER JOIN atau biasa disebut right join

1. LEFT JOIN

Pada LEFT JOIN output/hasil yang ditampilkan adalah data-data dari semua tabel yang terlibat baik yang hanya yang memiliki kondisi kesamaan data berdasarkan relasinya (kesamaan data foreign key dengan primary key tabel yang diacu) maupun data-data yang tidak memiliki kesamaan data berdasarkan relasinya dari tabel sebelah kiri dari klausa LEFT JOIN. Berikut adalah bentuk umum:

```
SELECT nm_tabel1.nm_kolom1, nm_tabel1.nm_kolom2, nm_tabel2.nm_kolom1, nm_tabel2.nm_kolom2 FROM tabel1 LEFT JOIN tabel2 ON tabel1.nama_kolom1(primary key)=tabel2.nama_kolom1(foreign key yg mengacu ke tabel1) WHERE kondisi
```

- Contoh :

```
select a.cust_name,b.order_date from customers a left join orders b on a.cust_id=b.cust_id RIGHT JOIN
```

Pada RIGHT JOIN output/hasil yang ditampilkan adalah data-data dari semua tabel yang terlibat baik yang hanya yang memiliki kondisi kesamaan data berdasarkan relasinya (kesamaan data foreign key dengan primary key tabel yang diacu) maupun data-data yang tidak memiliki kesamaan data berdasarkan relasinya dari tabel sebelah kanan dari klausa RIGHT JOIN.

```
SELECT nm_tabel1.nm_kolom1, nm_tabel1.nm_kolom2, nm_tabel2.
```

nm_kolom1, nm_tabel2.nm_kolom2 FROM tabel1 RIGHT JOIN tabel2 ON tabel1.nama_kolom1(primary key)=tabel2.nama_kolom1 (foreign key yg mengacu ke tabel1) WHERE kondisi;

- Contoh:

```
select a.cust_name,b.order_date from customers a right join orders  
b on a.cust_id=b.cust_id
```

2. SELF JOIN

Self join adalah melakukan join dengan dirinya sendiri. Atau join dengan table yang sama.

- Sintak nya sbb:

```
select nama alias1_table.kolom1, nama alias2_table.kolom2, from  
table alias1 inner join table alias2 on alias1.kolom3=alias2.kolom3
```

- Contoh:

```
select a.vend_name,b.vend_state, 'negaranya' ,b.vend_country from  
vendors a inner join vendors b on a.vend_id=b.vend_id
```

2.9 SubQuery

Subquery merupakan query didalam query. Umumnya, subquery ini dipakai untuk mencari data yang belum diketahui. Penggunaan query didalam query ini umumnya menjadi bagian dari kondisi. Sintak subquery adalah sebagai berikut:

```
SELECT <select_list>  
FROM <tabel>  
WHERE <column> =  
      (SELECT <single_column>  
       FROM <tabel>  
       WHERE <kondisi>);
```

2.10 Tipe Data

Dalam Database Data Type adalah suatu fungsi (function) yang digunakan untuk mengidentifikasi batasan suatu kolom dalam menyimpan dan penulisan format suatu data atau konten tertentu. Penggunaan typedata pada database memiliki beberapa fungsi yaitu : Untuk memberikan batasan atau

format pada kolom table suatu database.

Ada lima jenis tipe data sesuai dengan SQL - ANSI 1993 yaitu character string, numeric, temporal, binary, dan boolean.

a. Character String

Atribut seperti nama dan alamat direpresentasikan oleh character string.

Ada 2 macam tipe data untuk merepresentasikan character string, yaitu:

- CHARACTER() CHAR() menspesifikasikan karakter dengan panjang yang tetap. Sisa karakter yang tidak terpakai umumnya digantikan oleh padding characters (spasi).
- CHARACTER VARYING() Atau VARCHAR() menspesifikasikan karakter dengan panjang yang fleksibel dan maksimum sesuai dengan .

Sintak : Type Data [(M)]

Misal : CHAR [(M)] VARCHAR [(M)]

```
CREATE TABLE contoh_cha (cha CHAR(5), varcha VARCHAR(5));  
INSERT INTO contoh_cha values ('a ', 'a ');  
INSERT INTO contoh_cha values ('dunia', 'dunia');  
INSERT INTO contoh_cha VALUES ('basisdata', 'basisdata');
```

b. Numeric

Numeric, dari namanya sudah pasti numeric berarti digunakan pada kolom yang menyimpan data berupa angka. Tipe Data numeric memiliki beberapa format penulisan mislakan bilangan desimal, bilangan bulat, dil. Berikut ini beberapa contoh format dari tipe data numeric:

Type Data	Fungsi	Jangkuan / Range
INT	Menyimpan data dalam bentuk <i>integer</i> atau bilangan bulat dapat bernilai positif atau negatif	-2147483648 s/d 2147483647
TINYINT		-128 s/d 127
SMALLINT		-32.768 s/d 32.767
MEDIUMINT		-8.388.608 s/d 8.388.607
BIGINT		-9223372036854775808 s/d 9223372036854775807
FLOAT	Menyimpan data	3.402823466E+38 s/d -

	bilangan pecahan positif atau negatif	1.175494351E-38, 0, dan 1.175494351E-38 s/d 3.402823466E+38
DOUBLE		-1.79...E+308 s/d -2.22...E+308, 0, dan 2.22...E-308 s/d 1.79...E+308
DECIMAL/ NUMERIC		-1.79...E+308 s/d -2.22...E+308, 0, dan 2.22...E-308 s/d 1.79...E+308

Sintak : Type Data [(M[,D])] [UNSIGNED] [ZEROFILL]

Misal : INT / DECIMAL / FLOAT / DOUBLE[(M)] [UNSIGNED] [ZEROFILL]

Keterangan :

- M : menunjukkan lebar karakter maksimum, jumlah digit keseluruhan
- D : Jumlah digit dibelakang koma
- tanda [dan] berarti pemakaiannya adalah optional

Contoh :

```
CREATE TABLE contoh_int (mini TINYINT, kecil
SMALLINT UNSIGNED, sedang MEDIUMINT(4) ZEROFILL,
biasa INT(4) UNSIGNED, besar BIGINT(6) UNSIGNED
ZEROFILL);
```

c. Temporal

Temporal merupakan tipe data yang menyimpan tanggal dan waktu yang disesuaikan dengan system-timezone (komputer). Sebagai contoh data temporal adalah data tentang tanggal lahir. Ada dua macam tipe data temporal, yaitu:

- DATETIME. Tipe data ini menyimpan informasi tanggal, waktu atau bahkan keduanya. Dalam SQL Server, tipe data ini menyimpan dengan tingkat akurasi sampai 3,33 milidetik. Sedangkan untuk SMALLDATETIME hanya sampai 1 menit. Dalam tipe data ini, juga terdapat tipe data TIMESTAMP dengan tingkat akurasi sampai dengan 9 digit.

- **INTERVAL.** Umumnya digunakan untuk menyimpan periode seperti garansi. Ada 2 macam yaitu (1) YEAR-MONTH dan (2) DAY-TIME. SQL Server tidak mempunyai tipe data ini.

Tipe Data	Jangkauan	Ukuran	Zero Value
DATE	□1000-01-01□ to □9999-12-31□	3 byte	□0000-00-00□
DATETIME	□1000-□01-01 00:00:01□ to □9999-12-31 23:59:59□	8 byte	□0000-00-00 00:00:00□
TIMESTAMP	□1970-01-01 00:00:00□ to □2038-01-18 22:14:07□	4 byte	□0000-00-00 00:00:00□
TIME	□□838:59:59□ to □838:59:58□	3 byte	□00:00:00□
YEAR(2)	00 to 99	1 byte	□00□
YEAR(4)	1901 to 2155	1 byte	□0000□

Sintak : type data;

Contohnya : CREATE TABLE contoh_date (dat DATE, tim TIME, dattim DATETIME,timestam TIMESTAMP, yea YEAR);

Atau SELECT NOW(), CURDATE(), CURTIME();

2.11 Sintaks Stored Procedure

```

<create procedure statement> ::=
CREATE PROCEDURE <procedure name> ( [ <parameter list> ] )
<routine body>
<parameter list> ::=
<parameter specification> [ , <parameter specification> ]...
<parameter specification> ::=
[ IN | OUT | INOUT ] <parameter> <data type>
<routine body> ::= <begin-end block>
<begin-end block> ::=
[ <label> : ] BEGIN <statement list> END [ <label> ]
<statement list> ::= { <body statement> ; }...
<statement in body> ::=
<declarative statement> | <procedural statement>

```

a. Aktivasi/pemanggilan Stored Procedure:

```
<call statement> ::=  
CALL [ <database name> . ] <stored procedure name>  
( [ <scalar expression> [ , <scalar expression> ]... ] )
```

b. Menghapus Stored Procedure:

```
<drop procedure statement> ::=  
DROP PROCEDURE [ IF EXISTS ]  
[ <database name> . ] <procedure name>
```

c. Pernyataan pembuatan stored procedure :

```
DELIMITER $$ CREATE PROCEDURE set_counter(INOUT count  
INT(4),IN inc INT(4))  
BEGIN  
SET count = count + inc;  
END$$  
DELIMITER ;
```

d. Cara memanggilnya dengan mengset isi awal :

```
SET @counter = 1;  
CALL set_counter(@counter,1); -- 2  
CALL set_counter(@counter,2); -- 3  
CALL set_counter(@counter,9); -- 12  
SELECT @counter; -- 12
```

NUR H. H.

BAB III

TUGAS PENDAHULUAN

3.1 Soal

1. Jelaskan secara mendetail bagaimana penggunaan tipe data yang tepat dalam mendefinisikan tabel dapat mempengaruhi kinerja dan efisiensi basis data. Sertakan contoh kasus dimana pemilihan tipe data yang salah dapat menyebabkan masalah integritas data atau performa query. Dalam Penjelasan, Anda sertakan Pertimbangan yang harus diperhatikan dalam memilih tipe data untuk berbagai jenis kolom seperti karakter numerik, dan temporal.
2. Berikan analisis mendalam tentang bagaimana penggunaan stored procedure dapat meningkatkan keamanan dan konsistensi data dalam Sistem manajemen basis data. Jelaskan dengan detail bagaimana stored procedure dapat digunakan untuk membatasi akses dan manipulasi data oleh Pengguna yang berbeda. Sertakan contoh konkret dari kasus implementasi dimana stored procedure digunakan untuk memastikan bahwa hanya data yang memenuhi syarat tertentu yang dapat diubah.
3. Diskusikan berbagai teknik optimasi query yang dapat diterapkan pada database yang komplek dengan jumlah data yang sangat besar. Fokuskan pada Penggunaan indeks, Penggunaan tabel (join), dan subquery. Jelaskan bagaimana masing-masing teknik ini dapat meningkatkan kinerja query, dan sertakan contoh situasi dimana Penggunaan teknik-teknik ini diperlukan. Selain itu, analisis bagaimana struktur tabel dan relasi antar tabel dapat diatur untuk mendukung optimasi query yang efisien.

3.2 Jawaban

1. Pemilihan tipe data yang tepat sangat Penting dalam mendefinisikan tabel di database karena mempengaruhi Kinerja, efisiensi Pengiriman, dan integritas data. Tipe data yang sesuai dapat mengoptimalkan Pemrosesan query, Sedangkan tipe data yang salah dapat menyebabkan data korup

- Normalisasi : mengurangi redundansi dan memastikan data konsisten dengan memecah tabel yang lebih kecil dan terhubung dengan relasi yang sesuai
- Denormalisasi : mengurangi normalisasi untuk meningkatkan kinerja dengan mengurangi jumlah JOIN yang diperlukan
- Partisi Tabel : memecah tabel besar menjadi bagian-bagian yang lebih kecil untuk meningkatkan kinerja query.

Dengan menggunakan indeks, penggabungan tabel yang efisien, dan subquery yang tepat, serta mengoptimalkan struktur tabel, kinerja query dapat ditingkatkan

BAB IV

IMPLEMENTASI

4.1 Soal

1. Buatlah Stored Procedure INOUT, yang berguna untuk pencarian pengembalian berdasarkan tanggal yang kita inginkan dan hasilnya berupa keseluruhan data dalam tabel pengembalian yang sesuai dengan tanggal yang sudah diinputkan! Jika pada tabel pengembalian terdapat ID Anggota ataupun ID Petugas, maka tampilkan nama anggota & petugas berdasarkan ID tersebut!
 2. Buatlah stored procedure INOUT yang berguna untuk menampilkan daftar anggota berdasarkan status pinjam yang kita inginkan. Hasilnya berupa keseluruhan data dalam tabel anggota yang sesuai dengan status pinjam yang sudah diinputkan.
 3. Buatlah stored procedure dengan parameter OUT yang berfungsi untuk menampilkan daftar anggota berdasarkan status pinjam yang kita inginkan.
 4. Buatlah Stored Procedure dengan parameter IN yang berfungsi untuk menambahkan data baru ke tabel buku. Stored procedure ini harus menerima parameter tentang buku. Setelah data berhasil dimasukkan, stored procedure ini harus mengembalikan pesan konfirmasi bahwa data buku telah berhasil ditambahkan.
 5. Buatlah Stored Procedure dengan parameter IN yang berfungsi untuk menghapus data anggota berdasarkan ID Anggota. Stored procedure ini harus menerima parameter ID Anggota yang ingin dihapus dari tabel anggota. Setelah data berhasil dihapus, stored procedure ini harus return pesan konfirmasi bahwa data anggota telah berhasil dihapus. Selain itu, stored procedure ini juga harus memeriksa terlebih dahulu apakah anggota tersebut memiliki pinjaman yang belum dikembalikan, jika iya, proses penghapusan harus dibatalkan dan mengembalikan pesan kesalahan.
 6. Buatlah 3 Views, yang menggunakan fungsi JOIN, baik Right Join, Inner, dan Left. Pada tabel yang saling berelasi. Dan jelaskan perbedaannya. Serta Berikan tambahan agregasi dan pengelompokan dalam Views
-

tersebut, lalu analisislah hasilnya. Setiap Praktikan tidak diperbolehkan sama Studi Kasusnya!

4.2 Source Code

```
/Nomor 1/
DELIMITER //

CREATE PROCEDURE CariPengembalian(
    INOUT tanggal_kembali DATE
)
BEGIN
    SELECT
        p.kode_kembali,
        a.nama_anggota,
        b.judul_buku,
        t.nama AS nama_petugas,
        p.tgl_pinjam,
        p.tgl_kembali,
        p.denda
    FROM
        pengembalian p
    JOIN
        anggota a ON p.id_anggota = a.id_anggota
    JOIN
        buku b ON p.kode_buku = b.kode_buku
    JOIN
        petugas t ON p.id_petugas = t.id_petugas
    WHERE
        DATE(p.tgl_kembali) = tanggal_kembali;
END //

DELIMITER ;

SET @tanggal = '2024-05-13';
CALL CariPengembalian(@tanggal);

/Nomor 2/
DELIMITER //

CREATE PROCEDURE CariAnggotaBerdasarkanStatus(
    INOUT status_pinjam VARCHAR(15)
)
BEGIN
    SELECT
        id_anggota,
        nama_anggota,
        tempat_lahir_anggota,
        tanggal_lahir_anggota,
        no_telp,
        jenis_kelamin,
        status_pinjam
    FROM
        anggota
    WHERE
        status_pinjam = status_pinjam;
END //

DELIMITER ;

SET @status = 'pinjam';
CALL CariAnggotaBerdasarkanStatus(@status);
```

4.2 Hasil

4.2.1 Nomor 1

<input type="checkbox"/> kode_kembali	nama_anggota	judul_buku	nama_petugas	tgl_pinjam	tgl_kembali	denda
<input type="checkbox"/> PG001	Andi	Pelangi	Fitria	2024-05-06 20:25:16	2024-05-13 20:25:16	Rp.0
<input type="checkbox"/> PG002	Tomo	Bintang	Liana	2024-05-06 20:25:16	2024-05-13 20:25:16	Rp.0
<input type="checkbox"/> PG003	Rina	Bulan	Dandi	2024-05-06 20:25:16	2024-05-13 20:25:16	Rp.30000
<input type="checkbox"/> PM014	Hadi	Bintang	Dandi	2024-05-06 20:25:16	2024-05-13 20:25:16	RP.0
<input type="checkbox"/> PM015	Rina	Bulan	Dandi	2024-05-06 20:25:16	2024-05-13 20:25:16	Rp.30000

4.2.2 Nomor 2

<input type="checkbox"/> id_anggota	nama_anggota	tempat_lahir_anggota	tanggal_lahir_anggota	no_telp	jenis_kelamin	status_pinjam
<input type="checkbox"/> A001	Tomo	Jakarta	2004-06-22	2147483647	Laki-Laki	pinjam
<input type="checkbox"/> A002	Siska	Surabaya	2002-03-15	2147483647	Perempuan	pinjam
<input type="checkbox"/> A003	Budi	Bandung	2001-11-08	2147483647	Laki-Laki	pinjam
<input type="checkbox"/> A004	Dewi	Yogyakarta	2002-09-30	2147483647	Perempuan	pinjam
<input type="checkbox"/> A005	Hadi	Gresik	2003-07-12	2147483647	Laki-Laki	pinjam
<input type="checkbox"/> A006	Rina	Lamongan	2005-05-24	2147483647	Perempuan	pinjam
<input type="checkbox"/> A007	Adi	Bandung	2001-01-18	2147483647	Laki-Laki	pinjam
<input type="checkbox"/> A008	Lina	Jombang	2000-08-03	2147483647	Perempuan	pinjam
<input type="checkbox"/> A009	Andi	Bojonegoro	2004-04-06	2147483647	Laki-Laki	pinjam

4.2.3 Nomor 3

<input type="checkbox"/> id_anggota	nama_anggota	tempat_lahir_anggota	tanggal_lahir_anggota	no_telp	jenis_kelamin	status_pinjam
<input type="checkbox"/> A010	Rani	Sidoarjo	2004-12-29	2147483647	Perempuan	kembali

<input type="checkbox"/> keterangan
<input type="checkbox"/> Data buku dengan kode B011 berhasil ditambahkan.

4.2.4 Nomor 4

<input type="checkbox"/> result
<input type="checkbox"/> Tidak dapat menghapus anggota karena masih memiliki pinjaman yang belum di

4.2.5 Nomor 5

<input type="checkbox"/> nama_anak	merk_hp
<input type="checkbox"/> Ahmad	Samsung
<input type="checkbox"/> Budi	iPhone
<input type="checkbox"/> Cinta	(NULL)
<input type="checkbox"/> Dinda	(NULL)
<input type="checkbox"/> Erika	Xiaomi

4.2.6 Nomor 6

<input type="checkbox"/> nama_anak	merk_hp
<input type="checkbox"/> Ahmad	Samsung
<input type="checkbox"/> Budi	iPhone
<input type="checkbox"/> Erika	Xiaomi

4.3 Penjelasan

4.3.1 Nomor 1

Prosedur CariPengembalian digunakan untuk mencari data pengembalian buku berdasarkan tanggal pengembalian yang diberikan. Prosedur ini akan mengembalikan informasi seperti kode pengembalian, nama anggota, judul buku, nama petugas, tanggal pinjam, tanggal kembali, dan denda. Inputnya adalah tanggal pengembalian yang diterima sebagai parameter. Prosedur ini dijalankan dengan mengatur tanggal pengembalian terlebih dahulu dan kemudian memanggil prosedur dengan CALL CariPengembalian(@tanggal);.

4.3.2 Nomor 2

Prosedur CariAnggotaBerdasarkanStatus digunakan untuk mencari anggota berdasarkan status peminjaman mereka, seperti apakah mereka sedang meminjam buku atau tidak. Inputnya adalah status peminjaman yang diterima sebagai parameter. Prosedur ini akan mengembalikan informasi anggota seperti ID anggota, nama, tempat lahir, tanggal lahir, nomor telepon, jenis kelamin, dan status pinjam. Untuk menjalankan prosedur ini, status pinjam diatur terlebih dahulu dan kemudian prosedur dipanggil dengan CALL CariAnggotaBerdasarkanStatus(@status);.

4.3.3 Nomor 3

Prosedur daftarAnggota digunakan untuk menghitung dan menampilkan jumlah anggota yang memiliki status peminjaman tertentu. Inputnya adalah status peminjaman dan outputnya adalah jumlah anggota dengan status tersebut, serta menampilkan data lengkap anggota jika ada yang memenuhi kriteria tersebut. Prosedur ini dijalankan dengan memanggil CALL daftarAnggota(@agt, 'kembali');, dimana @agt adalah variabel yang akan menyimpan hasil hitungan anggota.

4.3.4 Nomor 4

Prosedur TambahBuku digunakan untuk menambahkan buku baru ke dalam database, asalkan buku dengan kode yang sama belum ada. Inputnya adalah kode buku, judul buku, pengarang, tahun terbit, dan jumlah buku. Jika buku

dengan kode tersebut sudah ada, prosedur akan memberikan pesan bahwa data buku sudah ada; jika belum ada, buku akan ditambahkan dan pesan konfirmasi keberhasilan akan ditampilkan. Prosedur ini dijalankan dengan memanggil CALL TambahBuku('B011', 'masih pemula', 'saya', '2024', '100');

4.3.5 Nomor 5

Prosedur HapusAnggota digunakan untuk menghapus data anggota dari database jika anggota tersebut tidak memiliki pinjaman yang belum dikembalikan. Inputnya adalah ID anggota. Prosedur ini akan mengecek jumlah pinjaman anggota dan jika masih ada pinjaman yang belum dikembalikan, akan menampilkan pesan bahwa anggota tidak dapat dihapus; jika tidak ada, anggota akan dihapus dan pesan konfirmasi keberhasilan akan ditampilkan. Prosedur ini dijalankan dengan memanggil CALL HapusAnggota('A001');

4.3.6 Nomor 6

membuat dua tabel, anak dan hp, kemudian mengisi data ke dalam tabel-tabel tersebut. Setelah itu, tiga jenis query JOIN digunakan untuk menampilkan data gabungan antara tabel anak dan hp: LEFT JOIN untuk menampilkan semua data anak dan merk HP, termasuk anak yang tidak memiliki HP; RIGHT JOIN untuk menampilkan semua data HP dan nama anak, termasuk HP yang tidak dimiliki oleh anak; dan INNER JOIN untuk menampilkan hanya data anak yang memiliki HP dan sebaliknya. Contoh query yang digunakan adalah SELECT anak.nama_anak, hp.merk_hp FROM modul5.anak LEFT JOIN modul5.hp ON anak.hp_id = hp.hp_id;

BAB V

PENUTUP

5.1 Analisa

Dari hasil praktikum sistem manajemen basis data, praktikan menganalisis bahwa praktikum ini memberikan pemahaman yang mendalam tentang penggunaan bahasa kueri SQL dalam mengelola data. Salah satu temuan utama adalah bahwa penggunaan perintah **SELECT** memungkinkan praktikan untuk mengambil data dari satu atau beberapa tabel dengan cara yang efisien. Selain itu, praktikan juga menemukan bahwa kemampuan untuk menyaring data menggunakan klausa **WHERE** memungkinkan mereka untuk mengidentifikasi dan memperoleh data yang sesuai dengan kriteria tertentu secara akurat.

Selama praktikum, penggunaan perintah pengurutan data dengan klausa **ORDER BY** membantu praktikan dalam mengurutkan hasil kueri sesuai dengan kebutuhan, baik secara naik maupun turun. Analisis juga menyoroti penggunaan operator **BETWEEN**, **IN**, dan **LIKE** yang memperluas fleksibilitas dalam pemilihan data berdasarkan rentang nilai, nilai tertentu dalam daftar, atau pola string yang cocok. Selain itu, praktikan mencatat bahwa penggunaan ekspresi kueri memberikan kemampuan tambahan dalam memanipulasi data dengan cara yang kompleks dan terstruktur.

5.2 Kesimpulan

1. Praktikum ini memberikan pemahaman yang mendalam tentang bahasa kueri SQL, termasuk penggunaan perintah **SELECT**, klausa **WHERE**, dan penggunaan berbagai operator seperti **BETWEEN**, **IN**, dan **LIKE**, yang merupakan fondasi penting dalam mengakses dan mengelola data dalam basis data.
 2. Dengan penerapan fungsi agregasi dan pengelompokan data, serta kemampuan untuk melakukan pengurutan data dan menyaring data berdasarkan kriteria tertentu, praktikan telah mengembangkan keterampilan analisis data yang lebih luas dan mendalam.
-

LAPORAN RESMI
MODUL VI
STORED PROCEDURE : BRANCING / LOOPING
SISTEM MANAJEMEN BASIS DATA



NAMA	: FAIRUZ ABDULLAH
N.R.P	: 220441100070
DOSEN	: MOHAMMAD SYARIEF, ST., MCs
ASISTEN	: NURI HIDAYATULOH
TGL PRAKTIKUM	: 03 MEI 2024

Disetujui : 16 MEI 2024
Asisten

NURI HIDAYATULOH
210441100100



LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I

PENDAHULUAN

1.1 Latar Belakang

Latar belakang dari praktikum "Stored Procedure: Branching/Looping pada SQL" adalah untuk memberikan pemahaman yang mendalam tentang penggunaan stored procedure dalam SQL, khususnya dalam konteks branching dan looping. Stored procedure adalah kumpulan pernyataan SQL yang telah disimpan di database dan dapat dipanggil dengan nama tertentu. Branching dan looping merupakan konsep penting dalam pemrograman yang memungkinkan alur eksekusi program untuk bercabang (branching) dan berulang (looping) sesuai dengan kondisi atau iterasi yang ditentukan. Dalam konteks SQL, penggunaan branching dan looping dalam stored procedure dapat meningkatkan fleksibilitas, efisiensi, dan keamanan dalam pengelolaan dan manipulasi data di dalam database. Praktikum ini bertujuan untuk memberikan pemahaman mendalam tentang konsep, sintaksis, dan penerapan branching dan looping dalam stored procedure SQL guna mendukung pengembangan aplikasi database yang lebih kompleks dan efisien.

1.2 Tujuan

- Mengerti tentang Stored Procedure
 - Mampu Menggunakan Variabel, Brancing/Looping serta Percabangan pada Stored Procedure
-

BAB II

DASAR TEORI

2.1 Dasar Teori

2.1.1 Variabel

Dideklarasikan dengan keyword “DECLARE” kemudian diikuti dengan nama variable dan tipe data.

Sintaks:

DECLARE namavariabel TYPE DEFAULT nilai;

- Untuk DEFAULT sifatnya opsional

DELIMITER //

CREATE PROCEDURE dec1()

BEGIN

DECLARE COUNT INT DEFAULT 0;

DECLARE x INT;

DECLARE message VARCHAR(100);

END;

DELIMITER ;

Contoh Cetak Nama :

DELIMITER |

CREATE PROCEDURE Excetak_nm(IN namadepan VARCHAR(20),

IN namabelakang VARCHAR(30), IN gelardepan CHAR (6),

IN gelarbalakang CHAR(10))

BEGIN

DECLARE nama VARCHAR(50);

SET nama =CONCAT (gelardepan, " ",namadepan, " ", namabelakang, " ",
gelarbelakang);

SELECT nama AS hasil;

END;

DELIMITER ;

Contoh prosedur untuk mencetak nama :

`call cetak_nm("Doni","Abdul","Dr.", "S.Kom, M.Kom")` standar di SQL. Pernyataan di antara BEGIN dan END merupakan badan (body) stored procedure. Perintah DELIMITER di akhir baris digunakan untuk mengembalikan delimiter ke karakter semula.

2.1.2. Pencabangan dan Pengulangan

[1] Percabangan

Penggunaan pernyataan-pernyataan pencabangan ataupun pengulangan di dalam stored procedure merupakan tindakan yang legal. Dengan demikian, kita bisa menghasilkan suatu prosedur yang kompleks. Perintah pemilihan ini berupa statement-statement yang akan mengerjakan instruksi jika kondisi benar/terpenuhi.

Contoh berikut memperlihatkan penggunaan pernyataan IF

Sintak :

```
IF [val] THEN IF [val] THEN
[result1] [result1] END IF ; ELSE
[result2] END IF ;
```

DELIMITER //

```
CREATE PROCEDURE cobaIF(
IN bil INT(3)
)
BEGIN
/*Deklarasi Variabel*/ DECLARE
str VARCHAR(50); if (bil<0) then
SET str ='Bilangan Negetif'; ELSE
```

```

SET str='Bilangan Posistif';
END if;
    SELECT str; END//
DELIMITER;

```

Atau

```

DELIMITER |
CREATE PROCEDURE exIF(
IN bil INT(3)
)
BEGIN
/*Deklarasi Variabel*/ DECLARE
str VARCHAR(50); if (bil<0) then
    SET str ='Bilangan Negetif'; ELSE
SET str='Bilangan Posistif';
END if;
SELECT str;
END;

```

Apabila pemanggilan stored procedure di atas mengabaikan argumen, DBMS akan merespon dengan pesan kesalahan. Bergantung kebutuhan, pendefinisian parameter pada stored procedure juga bisa lebih dari satu. Sebagai contoh, buat stored procedure dengan dua buah parameter

```

DELIMITER //
CREATE PROCEDURE getNamaByAlmtJKelamin(
    IN Alamat VARCHAR(20),
    IN jeniskelamin VARCHAR(2))
BEGIN
    SELECT * FROM mhs
    WHERE jklmn = jeniskelamin
    AND almtMhs = Alamat ;
END //
DELIMITER ;

```

seperti berikut :

Pemanggilan stored procedure di atas tentunya akan memerlukan dua buah

argumen.

```
CALL getNamaByAlmtJKelamin ('surabaya','L');
```

idMhs	namaMhs	almtMhs	jKlms
1011	Ari	Surabaya	L

Penambahan Data

DELIMITER ;

Untuk memanggilnya :

```
call exIF(7);
```

[2] Pengulangan

Perintah perulangan dengan menggunakan statement LOOP, WHILE, dan REPEAT., Penggunaan statement LOOP diawali dengan menentukan nama perulangan : LOOP dan diakhiri dengan END LOOP.

Statement WHILE Statement WHILE melakukan perulangan berdasarkan kondisi tertentu. Perulangan akan dilakukan jika kondisi bernilai benar/true.

Sintak Loop :

Loop_name : LOOP

[statement1]

[statement2]

END LOOP loop_name

Sintak While :

Loop_Name : WHILE [condition] DO

[statement1]

[statement2]

END WHILE Loop_Name;

Sintak Repeat :

```
[begin_label:]  
REPEAT  
statement_list UNTIL  
search_condition  
END REPEAT [end_label]
```

Contoh penggunaan looping adalah sebagai berikut

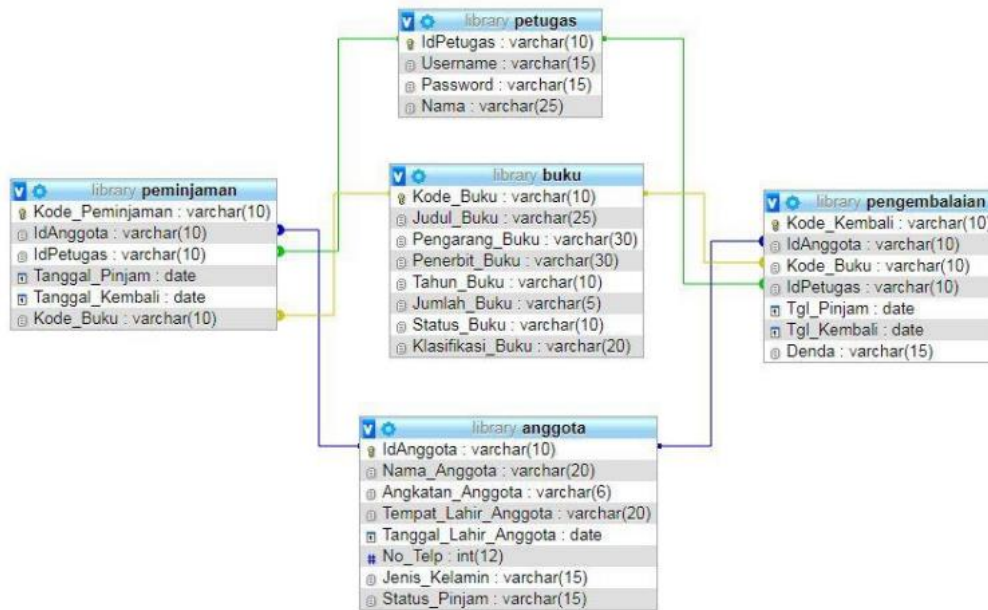
```
DELIMITER |  
CREATE PROCEDURE ExLoop(  
IN bil INT(3)  
)  
BEGIN  
/*Deklarasi Variabel*/  
DECLARE str VARCHAR(150);  
DECLARE i int(3); Set i=1;  
Set str=, ;
```

```
While i<= bil do  
Set str=concat (str, i, , );  
Set i=i+1;  
END WHILE;  
SELECT str;  
END;
```

BAB III

IMPLEMENTASI

3.1 Soal



- 1) Buatlah Stored Procedure Variabel untuk menampilkan Biodata masing-masing dengan isi : Nim, Nama mahasiswa, Alamat, No Telp, jenis Kelamin dan Hobi serta tambahkan umur sekarang. Kemudian tampilkan hasilnya.
- 2) Buatlah Stored Procedure untuk menentukan keterangan pengingat pengembalian buku, dengan ketentuan jika tanggal pinjam ≤ 2 hari “Silahkan penggunaan buku dengan baik”, jika tanggal pinjam antara 3-5 hari “Ingat!, Waktu pinjam segera habis”, dan jika tanggal pinjam ≥ 6 hari “Warning!!!, Denda menanti anda!”.
- 3) Buatlah Stored Procedure untuk memeriksa jumlah denda yang dimiliki mahasiswa! Jika mahasiswa mendapat denda maka akan menampilkan data denda yang belum dibayarkan, jika mahasiswa sudah membayar atau tidak memiliki tanggungan denda maka akan menampilkan pesan bahwa mahasiswa tersebut tidak memiliki tanggungan atau denda.

- 4) Buatlah Stored procedure Looping untuk mencetak data peminjaman 1 sampai dengan 10.
 - 5) Hapuslah anggota dengan jenis kelamin Laki-laki dari basisdata PERPUSTAKAAN. Akan tetapi jika Anggota tersebut mempunyai status pinjam tidak nol, maka pemain tidak boleh dihapus dari basisdata!
-

3.2 Source Code

```
CREATE DATABASE modul4;
USE modul4;

CREATE TABLE IF NOT EXISTS anggota (
id_anggota VARCHAR(10) NOT NULL PRIMARY KEY,
nama_anggota VARCHAR(20) NOT NULL,
tempat_lahir_anggota VARCHAR(20) NOT NULL,
tanggal_lahir_anggota DATE,
no_telp INT(12) NOT NULL,
jenis_kelamin VARCHAR(15) NOT NULL,
status_pinjam VARCHAR(15) NOT NULL
);

CREATE TABLE IF NOT EXISTS petugas (
id_petugas VARCHAR(10) NOT NULL PRIMARY KEY,
username VARCHAR(15) NOT NULL,
PASSWORD VARCHAR(15) NOT NULL,
nama VARCHAR(25) NOT NULL
);

CREATE TABLE IF NOT EXISTS buku (
kode_buku VARCHAR(10) NOT NULL PRIMARY KEY ,
judul_buku VARCHAR(25) NOT NULL,
pengarang_buku VARCHAR(30) NOT NULL,
tahun_buku VARCHAR(10) NOT NULL,
jumlah_buku VARCHAR(5) NOT NULL
);

CREATE TABLE IF NOT EXISTS peminjaman (
kode_peminjaman VARCHAR (10) NOT NULL PRIMARY KEY,
id_anggota VARCHAR (10) NOT NULL,
id_petugas VARCHAR (10) NOT NULL,
tanggal_pinjam TIMESTAMP DEFAULT NOW(),
tanggal_kembali TIMESTAMP DEFAULT NOW(),
kode_buku VARCHAR (10) NOT NULL,
FOREIGN KEY (id_anggota) REFERENCES anggota(id_anggota),
FOREIGN KEY (id_petugas) REFERENCES petugas(id_petugas),
FOREIGN KEY (kode_buku) REFERENCES buku (kode_buku)
);

CREATE TABLE IF NOT EXISTS pengembalian (
kode_kembali VARCHAR (10) NOT NULL PRIMARY KEY,
id_anggota VARCHAR (10) NOT NULL,
kode_buku VARCHAR (10) NOT NULL,
id_petugas VARCHAR (10) NOT NULL,
tgl_pinjam TIMESTAMP DEFAULT NOW(),
tgl_kembali TIMESTAMP DEFAULT NOW(),
denda VARCHAR (15) NOT NULL,
FOREIGN KEY (id_anggota) REFERENCES anggota(id_anggota),
FOREIGN KEY (kode_buku) REFERENCES buku(kode_buku),
FOREIGN KEY (id_petugas) REFERENCES petugas(id_petugas)
);

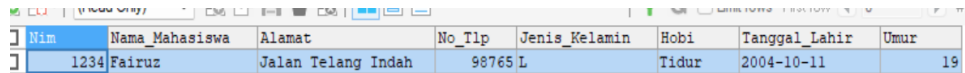
SHOW TABLES;

INSERT INTO anggota VALUES
('A001','Tomo','Jakarta','2004-06-22','085211112222','Laki-Laki','1'),
('A002','Siska','Surabaya','2002-03-15','085233334444','Perempuan','1'),
('A003','Budi','Bandung','2001-11-08','085255556666','Laki-Laki','0'),
('A004','Dewi','Yogyakarta','2002-09-30','085277778888','Perempuan','1'),
('A005','Hadi','Gresik','2003-07-12','085299990000','Laki-Laki','5'),
('A006','Rina','Lamongan','2005-05-24','085211112233','Perempuan','1'),
('A007','Adi','Bandung','2001-01-18','085211112244','Laki-Laki','1'),
('A008','Lina','Jombang','2000-08-03','085211112255','Perempuan','1'),
('A009','Andi','Bojonegoro','2004-04-06','085211112266','Laki-Laki','1'),
('A010','Rani','Sidoarjo','2004-12-29','085211112277','Perempuan','0');

UPDATE anggota
SET status_pinjam = '0'
WHERE id_anggota = 'A003';
```

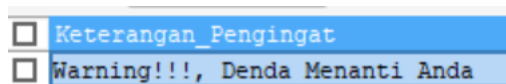
3.3 Hasil

3.3.1 Nomor 1



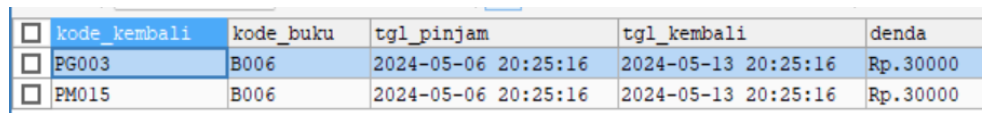
Nim	Nama_Mahasiswa	Alamat	No_Tlp	Jenis_Kelamin	Hobi	Tanggal_Lahir	Umur
1234	Fairuz	Jalan Telang Indah	98765	L	Tidur	2004-10-11	19

3.3.2 Nomor 2



Keterangan_Pengingat
Warning!!!, Denda Menanti Anda

3.3.3 Nomor 3



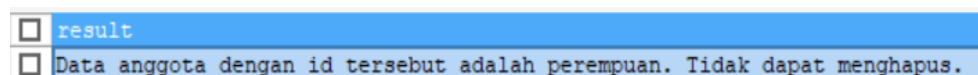
kode_kembali	kode_buku	tgl_pinjam	tgl_kembali	denda
PG003	B006	2024-05-06 20:25:16	2024-05-13 20:25:16	Rp.30000
PM015	B006	2024-05-06 20:25:16	2024-05-13 20:25:16	Rp.30000

3.3.4 Nomor 4



kode_peminjaman	id_anggota	id_petugas	tanggal_pinjam	tanggal_kembali	kode_buku
PM001	A004	P004	2024-05-06 20:22:02	2024-05-13 20:22:02	B001

3.3.5 Nomor 5



result
Data anggota dengan id tersebut adalah perempuan. Tidak dapat menghapus.

3.4 Penjelasan

3.4.1 Nomor 1

Prosedur ini menerima beberapa parameter masukan (NIM, nama, alamat, nomor telepon, jenis kelamin, hobi, dan tanggal lahir). Prosedur ini mendeklarasikan variabel-variabel untuk menyimpan nilai parameter, menghitung umur berdasarkan tanggal lahir, dan kemudian menampilkan informasi biodata mahasiswa beserta umurnya.

3.4.2 Nomor 2

Kedua prosedur ini menerima satu parameter masukan yaitu tanggal pinjam. Mereka menghitung selisih hari antara tanggal pinjam dan tanggal sekarang. Berdasarkan selisih ini, mereka memberikan pesan pengingat mengenai status peminjaman buku. Prosedur pertama menggunakan struktur IF-ELSEIF-ELSE untuk menentukan pesan pengingat,

sedangkan prosedur kedua menggunakan CASE statement.

3.4.3 Nomor 3

Prosedur ini mengecek apakah seorang mahasiswa memiliki denda yang belum dibayar berdasarkan `student_id`. Jika ada denda, prosedur ini menampilkan rincian peminjaman dan denda yang belum dibayar. Jika tidak ada denda, prosedur menampilkan pesan bahwa mahasiswa tidak memiliki tanggungan atau denda.

3.4.4 Nomor 4

Prosedur ini melakukan iterasi sebanyak 10 kali (dari $i=1$ sampai $i=10$), dan pada setiap iterasi, prosedur menampilkan data peminjaman berdasarkan `kode_peminjaman` yang dibentuk dari nilai iterasi. Ini menggunakan loop WHILE.

3.4.5 Nomor 5

Prosedur ini menghapus data anggota berdasarkan `anggota_id`. Sebelum menghapus, prosedur mengecek dua kondisi:

Apakah anggota tersebut memiliki pinjaman yang belum selesai jika jenis kelaminnya adalah laki-laki.

Apakah anggota tersebut adalah perempuan.

Jika salah satu kondisi terpenuhi, prosedur menampilkan pesan yang sesuai dan tidak menghapus data. Jika tidak, prosedur akan menghapus data anggota dan menampilkan pesan konfirmasi bahwa data berhasil dihapus.

BAB IV

PENUTUP

4.1 Analisa

Dari hasil praktikum, praktikan menganalisa bahwa penggunaan stored procedure dengan konsep branching dan looping dalam SQL memberikan kemampuan yang lebih fleksibel dan efisien dalam pengelolaan data di dalam database. Dalam praktikum ini, praktikan mempelajari berbagai fitur dan sintaksis yang mendukung pembuatan stored procedure yang kompleks, termasuk penggunaan pernyataan IF-ELSEIF-ELSE untuk branching serta penggunaan pernyataan WHILE untuk looping. Analisis ini menggarisbawahi pentingnya pemahaman tentang logika percabangan dan pengulangan dalam SQL untuk mengoptimalkan kinerja dan fleksibilitas proses database. Dengan menggunakan stored procedure yang memanfaatkan branching dan looping, praktikan dapat mengimplementasikan logika bisnis yang kompleks, melakukan pengolahan data yang lebih efisien, dan meningkatkan kinerja aplikasi database secara keseluruhan.

4.2 Kesimpulan

1. Praktikum stored procedure dengan branching dan looping memberi saya pemahaman yang lebih dalam tentang bagaimana logika program dieksekusi di dalam database.
 2. Dengan memahami stored procedure, saya menyadari pentingnya penggunaannya dalam mengoptimalkan kinerja aplikasi.
 3. stored procedure memungkinkan saya untuk memisahkan logika bisnis dari kode aplikasi klien.
 4. Dengan menggunakan stored procedure, kami dapat meningkatkan keamanan database dengan mengontrol akses ke data melalui prosedur yang telah ditentukan.
-

LAPORAN RESMI
MODUL VII
TRIGGER
SISTEM MANAJEMEN BASIS DATA



NAMA	: FAIRUZ ABDULLAH
N.R.P	: 220441100070
DOSEN	: MOHAMMAD SYARIEF, ST., M.Cs
ASISTEN	: NURI HIDAYATULOH
TGL PRAKTIKUM	: 24 MEI 2024

Disetujui 06 JUNI 2024
Asisten

NURI HIDAYATULOH
210441100100



LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam pengembangan sistem manajemen basis data (Database Management System, DBMS), trigger adalah salah satu fitur penting yang sering digunakan untuk menjaga integritas data dan menerapkan logika bisnis di tingkat basis data. Trigger adalah prosedur yang dijalankan secara otomatis sebagai respons terhadap peristiwa tertentu pada tabel atau tampilan dalam database, seperti operasi INSERT, UPDATE, atau DELETE. Fitur ini memungkinkan pengembang untuk memastikan bahwa aturan dan kebijakan yang diterapkan pada data selalu dipatuhi, tanpa memerlukan intervensi manual setiap kali terjadi perubahan data.

Pemahaman tentang trigger sangat penting bagi para mahasiswa dan pengembang perangkat lunak, karena trigger dapat digunakan untuk melakukan validasi data, menjaga konsistensi data, dan menjalankan proses otomatisasi yang kompleks. Misalnya, dalam sistem perbankan, trigger dapat digunakan untuk otomatisasi penghitungan bunga atau untuk memblokir transaksi yang mencurigakan. Dalam konteks lain, trigger juga bisa digunakan untuk mencatat log perubahan data, yang sangat penting dalam audit dan penelusuran jejak transaksi.

Melalui praktikum ini, mahasiswa diharapkan dapat memahami manfaat dan potensi masalah yang mungkin timbul dari penggunaan trigger, serta bagaimana mengatasi tantangan tersebut. Selain itu, pengalaman praktis ini akan memberikan wawasan yang lebih baik tentang bagaimana basis data dapat digunakan untuk mengotomatisasi tugas-tugas yang berulang dan menjaga integritas data, yang pada akhirnya akan meningkatkan kualitas dan performa aplikasi yang dikembangkan.

1.2 Tujuan

- Mengetahui trigger
 - Mampu mendesain trigger sesuai kebutuhan sistem
-

BAB II

DASAR TEORI

2.1 DASAR TEORI

Server database secara normal bersifat pasif. Database akan melakukan aksi ketika kita secara eksplisit memberikan perintah secara tertulis, misalnya melalui perintah SQL. Kita bisa menset agar database berubah dari passive menjadi aktif dengan menggunakan trigger.

Trigger adalah kode perintah SQL yang berisi perintah sql procedural dan perintah deklarative yang tersimpan di dalam database dan di aktifkan / dijalankan oleh server database jika sebuah operasi tertentu dijalankan didalam database.

MySQL akan menjalankan trigger ketika ada program, atau user, atau store procedur yang menjalankan perintah database tertentu, yaitu ketika menambahkan baris data ke tabel atau ketika menghapus semua data dari tabel. MySQL akan menjalankan trigger secara otomatis sesuai dengan kondisi tersebut. Trigger tidak bisa dipanggil atau di batalkan dari program.

Untuk mendefinisikan trigger menggunakan perintah CREATE TRIGGER, dengan diikuti dengan elemen trigger, yaitu :

- trigger moment (before / after)
- trigger event (insert, delete, update)
- dan trigger action (yang dilakukan)

Contoh

```
CREATE TRIGGER INSERT_PLAYERS
AFTER
INSERT ON PLAYERS FOR EACH ROW
BEGIN
    INSERT INTO CHANGES
        (USER, CHA_TIME, CHA_PLAYERNO,
        CHA_TYPE, CHA_PLAYERNO_NEW)
    VALUES (USER, CURDATE(), NEW.PLAYERNO, 'I', NULL);
END
```

Trigger juga bisa memanggil stored procedure, misalnya

```

CREATE PROCEDURE INSERT_CHANGE
  (IN CPNO      INTEGER,
   IN CTYPE     CHAR(1),
   IN CPNO_NEW  INTEGER)
BEGIN
  INSERT INTO CHANGES (USER, CHA_TIME, CHA_PLAYERNO,
                       CHA_TYPE, CHA_PLAYERNO_NEW)
  VALUES (USER, CURDATE(), CPNO, CTYPE, CPNO_NEW);
END

```

2 trigger tidak bisa memiliki momen yang sama dan event yang sama dalam 1 tabel. Sebagai contoh, kita tidak bisa membuat 2 trigger BEFORE DELETE di tabel 'mahasiswa'. Jika kita menginginkan ada 2 program untuk tabel tertentu, maka kita harus menggabungkannya dalam 1 trigger (bisa di pisah dituliskan dalam store procedure)

Ketika kita melakukan update record, ada 2 variabel yang muncul dalam sistem, NEW dan OLD. OLD menyimpan isi record dari data yang lama, dan NEW menyimpan isi record dari data yang baru. Kita bisa menggunakan 2 variabel ini di trigger.

Contoh

```

CREATE TRIGGER DELETE_PLAYER
  AFTER DELETE ON PLAYERS FOR EACH ROW
BEGIN
  CALL INSERT_CHANGE (OLD.PLAYERNO, 'D', NULL);
END

```

Trigger juga bisa dimanfaatkan untuk melakukan pengecekan integrity constraint dan pengecekan data yang akan disimpan ke dalam tabel.

Contoh

Alhamdulillah
NUP 21

BAB II

TUGAS PENDAHULUAN

3.1 Soal

1. Bagaimana operasi Delete berbeda dari Insert dan Update? Jelaskan dampak dari operasi Delete pada data yang ada di dalam tabel
2. Bagaimana Penggunaan konsep OLD dan NEW dapat mempengaruhi logika bisnis dalam Pengembangan Program? Berikan contoh skenario dimana Pemahaman tentang OLD dan NEW diperlukan untuk mengimplementasikan logika bisnis yang kompleks.

3.2 Jawaban

1. Operasi Delete berbeda dari Insert dan Update karena Delete untuk menghapus basis yang ada di tabel sedangkan Insert dan Update untuk menambah dan memodifikasi. Dampak dari operasi delete adalah penghapusan data secara permanen, dapat berdampak pada integritas referensi.
2. Trigger new dan old dapat digunakan untuk memonitoring data terkait secara otomatis, dapat juga untuk mencatat perubahan yang dilakukan dalam data.
Contoh skenario:
Misal kita punya sistem perbankan, kita bisa memasukkan saldo aktif nasabah tidak pernah negatif setelah transaksi. Kita dapat menggunakan trigger dengan konsep old dan new untuk mengimplementasikan logika ini.

BAB IV

IMPLEMENTASI

4.1 Soal

Desain trigger (dan store procedur sesuai kebutuhan) untuk

1. Pastikan `tgl_rencana_kembali` tidak lebih awal dari `tgl_pinjam`
 2. Ketika mobil dikembalikan, `tgl_kembali` di isi, juga menghitung `total_bayar` dan `denda`(jika ada)
 3. Ketika insert data ke tabel pelanggan, pastikan panjang NIK sesuai dengan aturan yang berlaku
 4. Ketika insert data ke tabel mobil, pastikan di kolom `platno`, 1/2 karakter awal harus huruf
-

4.2 Source Code

```
CREATE DATABASE modul7;
USE modul7

CREATE TABLE MOBIL (
    ID_MOBIL INT AUTO_INCREMENT PRIMARY KEY,
    PLATNO VARCHAR(15) NOT NULL,
    MERK VARCHAR(50) NOT NULL,
    JENIS VARCHAR(50) NOT NULL,
    HARGA_SEWA_PERHARI DECIMAL(10, 2) NOT NULL
);

CREATE TABLE PELANGGAN (
    ID_PELANGGAN INT AUTO_INCREMENT PRIMARY KEY,
    NAMA VARCHAR(100) NOT NULL,
    ALAMAT TEXT NOT NULL,
    NIK VARCHAR(20) NOT NULL,
    NO_TELEPON VARCHAR(15) NOT NULL,
    JENIS_KELAMIN ENUM('Laki-laki', 'Perempuan') NOT NULL
);

CREATE TABLE PEMINJAMAN (
    ID1 INT AUTO_INCREMENT PRIMARY KEY,
    ID_MOBIL INT NOT NULL,
    ID_PELANGGAN INT NOT NULL,
    TGL_PINJAM DATE NOT NULL,
    TGL_RENCANA_KEMBALI DATE NOT NULL,
    TOTAL_HARI INT NOT NULL,
    TOTAL_BAYAR DECIMAL(10, 2) NOT NULL,
    TGL_KEMBALI DATE,
    DENDA DECIMAL(10, 2),
    FOREIGN KEY (ID_MOBIL) REFERENCES MOBIL(ID_MOBIL),
    FOREIGN KEY (ID_PELANGGAN) REFERENCES PELANGGAN(ID_PELANGGAN)
);

INSERT INTO MOBIL (PLATNO, MERK, JENIS, HARGA_SEWA_PERHARI)
VALUES
('B 1234 CD', 'Toyota', 'SUV', 500000),
('B 5678 EF', 'Honda', 'Sedan', 400000);

INSERT INTO PELANGGAN (NAMA, ALAMAT, NIK, NO_TELEPON, JENIS_KELAMIN)
VALUES
('John Doe', 'Jl. Kebon Jeruk No. 1', '1234567890123456', '08123456789', 'Laki-laki'),
('Jane Doe', 'Jl. Kebon Anggrek No. 2', '1234567890123457', '08123456780', 'Perempuan');

INSERT INTO PEMINJAMAN (ID_MOBIL, ID_PELANGGAN, TGL_PINJAM, TGL_RENCANA_KEMBALI,
TOTAL_HARI, TOTAL_BAYAR, TGL_KEMBALI, DENDA)
VALUES
(1, 1, '2023-01-01', '2023-01-05', 4, 2000000, '2023-01-05', 0),
(2, 2, '2023-02-01', '2023-02-03', 2, 800000, '2023-02-04', 100000);

/*Nomor 1*/
SET sql_safe_updates = 0;

DELIMITER//
CREATE TRIGGER before_insert_peminjaman
BEFORE INSERT ON PEMINJAMAN
FOR EACH ROW
BEGIN
```

4.2 Hasil

4.2.1 Nomor 1

Error Code: 1644

Tanggal rencana kembali tidak boleh lebih awal dari tanggal pinjam

4.2.2 Nomor 2

ID_MOBIL	ID_PELANGGAN	TGL_PINJAM	TGL_RENCANA_KEMBALI	TOTAL_HARI	TOTAL_BAYAR	TGL_KEMBALI	DENDA
1	1	2023-01-01	2023-01-05	4	4500000.00	2023-01-10	2500000.00
2	2	2023-02-01	2023-02-03	2	800000.00	2023-02-04	100000.00
*	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

4.2.3 Nomor 3

Error Code: 1644

Panjang NIK harus 16 karakter

4.2.4 Nomor 4

Error Code: 1644

Karakter pertama pada PLATNO harus huruf

4.3 Penjelasan

4.3.1 Nomor 1

Jika tanggal rencana pengembalian yang dimasukkan lebih awal dari tanggal peminjaman, maka trigger ini akan menghasilkan sebuah pesan error dan mencegah operasi insert dilanjutkan. Dengan demikian, praktik ini membantu menghindari terjadinya kesalahan data yang mungkin timbul karena pengisian yang tidak sesuai dengan aturan yang telah ditetapkan

4.3.2 Nomor 2

Trigger ini bertujuan untuk menghitung total bayar dan denda yang harus dibayar oleh pelanggan berdasarkan tanggal pengembalian yang baru. Dengan menggunakan trigger ini, setiap kali terjadi pembaruan pada tanggal pengembalian, sistem akan secara otomatis menghitung ulang total bayar dan denda berdasarkan perbedaan tanggal pengembalian baru dengan tanggal rencana pengembalian yang tercatat sebelumnya. Ini memastikan bahwa informasi pembayaran selalu terupdate dan akurat sesuai dengan situasi terkini.

4.3.3 Nomor 3

Dua trigger ini ditambahkan untuk memvalidasi data yang dimasukkan ke dalam tabel PELANGGAN. Trigger pertama memeriksa panjang NIK yang dimasukkan, memastikan bahwa NIK memiliki panjang tepat 16 karakter. Jika panjang

NIK tidak sesuai, trigger akan menghasilkan pesan error dan mencegah operasi insert dilanjutkan. Sementara itu, trigger kedua melakukan pemeriksaan serupa untuk operasi update pada NIK. Keduanya bertujuan untuk memastikan bahwa data yang dimasukkan ke dalam sistem memenuhi standar yang telah ditetapkan.

4.3.4 Nomor 4

Trigger ini dirancang untuk memvalidasi data yang dimasukkan ke dalam tabel MOBIL, khususnya pada kolom PLATNO. Trigger akan memeriksa karakter pertama pada PLATNO, dan jika karakter tersebut bukan huruf, maka trigger akan menghasilkan pesan error dan mencegah operasi insert dilanjutkan. Dengan menggunakan trigger ini, sistem dapat memastikan bahwa data yang dimasukkan ke dalam tabel MOBIL sesuai dengan format yang telah ditentukan.

BAB V

PENUTUP

5.1 Analisa

Dari hasil praktikum sistem manajemen basis data, praktikan menganalisis bahwa penggunaan trigger dalam database sangat efektif untuk menjaga integritas dan konsistensi data. Trigger memungkinkan pelaksanaan otomatis dari aturan bisnis yang kompleks pada tingkat basis data, yang memastikan bahwa semua operasi pada tabel tertentu selalu mematuhi kebijakan yang telah ditentukan. Praktikan menemukan bahwa trigger dapat diimplementasikan dengan mudah untuk menangani berbagai skenario seperti validasi data saat operasi INSERT, memastikan bahwa perubahan tertentu pada data memicu pembaruan terkait secara otomatis saat operasi UPDATE, dan mengelola penghapusan data secara terkendali saat operasi DELETE.

Penggunaan trigger juga membantu mengurangi kesalahan manusia dengan mengotomatisasi tugas-tugas berulang yang sebelumnya perlu dilakukan secara manual. Misalnya, dalam situasi di mana log perubahan data perlu dicatat untuk tujuan audit, trigger dapat memastikan bahwa setiap perubahan pada data tercatat secara akurat tanpa memerlukan kode tambahan pada tingkat aplikasi. Praktikan mencatat bahwa dengan menggunakan trigger, efisiensi pengelolaan basis data dapat meningkat karena banyak proses yang dapat dijalankan secara otomatis di belakang layar.

5.2 Kesimpulan

1. Mahasiswa mampu memahami konsep dasar trigger insert, delete, dan update pada MySQL.
 2. Mereka dapat membuat trigger yang memvalidasi, mengontrol, dan mengelola data yang dimasukkan ke dalam tabel, serta menangani operasi delete dan update.
 3. Praktikum ini memberikan pemahaman yang kuat tentang penggunaan trigger untuk meningkatkan keamanan, integritas, dan efisiensi dalam pengelolaan basis data.
-

PENUTUP

A. Kesimpulan

Praktikum Sistem Manajemen Basis Data pada semester ini menurut penyusun memberikan pemahaman mengenai konsep-konsep dasar serta implementasi dalam pengelolaan basis data. Melalui praktikum ini, saya belajar untuk membuat, mengelola, dan memanipulasi basis data. Penulis juga mempelajari cara merancang skema basis data yang efisien, termasuk pembuatan tabel, relasi antar tabel, serta penerapan materi seperti view, stored procedure, branching, looping, dan trigger.

Pada view dan stored procedure kita dapat menyimpan hasil query kita, jadi kita cuman memanggilnya sewaktu-waktu tanpa membuat query lagi. Bedanya untuk view hanya bisa select sedangkan stored procedure lebih kompleks seperti memanipulasi database, kondisi dan looping begitu juga dengan 3 parameter yaitu In, Out, dan INOUT. Untuk trigger adalah fungsi yang akan berjalan otomatis selama hal itu memicu trigger tersebut, pada trigger kita juga memanipulasi database.

Secara keseluruhan, praktikum SMBD ini memberikan landasan yang kuat bagi peserta untuk memahami dan menerapkan konsep-konsep dasar dalam pengelolaan basis data, yang mungkin akan berguna dalam dunia nyata.

B. Saran

Menurut pendapat penyusun saran yang bisa saya berikan adalah semoga praktikum smbd menjadi lebih baik lagi seperti materi yang lebih mudah dipahami dan asprak-asprak yang lebih ahli dalam menjelaskan. Untuk fasilitas di lab mungkin kedepannya bisa lebih baik lagi karena proyektor sering bermasalah saat praktikum. Untuk asprak terima kasih telah meluangkan waktu untuk menjawab pertanyaan dan selalu bertanya permasalahan materi apa yang mungkin kurang saya pahami. Semoga ke depannya, kerjasama dan komunikasi antara mahasiswa dan asprak dapat lebih ditingkatkan.

BIOGRAFI PENYUSUN



Nama : Fairuz Abdullah
TTL : Surabaya, 11 Oktober 2004
Alamat : JL. Simolawang 6 / 22
Jurusan : Sistem Informasi
Universitas : Universitas Trunojoyo Madura
Hobi : Sketsa character
Cita-cita : Digital art
HP : 085607028624
e-Mail : fairuzabdullah0@gmail.com

RIWAYAT PENDIDIKAN

1. SDN SIMOKERTO I (2010-2016)
 2. SMPN 8 SURABAYA (2016-2019)
 3. SMA MUJAHIDIN SURABAYA (2019-2022)
 4. UNIVERSITAS TRUNOJOYO MADURA (2022-Sekarang)
-